

Oracle Utilities
Customer Care and Billing

Administrative User Guide

Release 2.5.0 Service Pack 2

E61802-03

July 2016

Oracle Utilities Customer Care and Billing Administrative User Guide

Release 2.5.0 Service Pack 2

E61802-03

July 2016

Documentation build: 6.15.2016 18:18:24 [C1_1465953504000]

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Administrative User Guides.....	22
Framework Administrative User Guide.....	22
Defining General Options.....	23
Defining Installation Options.....	23
Installation Options - Main.....	23
Installation Options - Messages.....	24
Installation Options - Algorithms.....	24
Installation Options - Accessible Modules.....	26
Installation Options - Installed Products.....	26
Support For Different Languages.....	26
User Language.....	27
Customer Language.....	27
Defining Languages.....	27
Defining Countries.....	28
Country - Main.....	29
Country - States.....	29
Defining Currency Codes.....	29
Defining Time Zones.....	30
Designing Time Zones.....	30
Setting Up Time Zones.....	30
Setting Up Seasonal Time Shift.....	31
Defining Geographic Types.....	31
Defining Work Calendar.....	32
Defining Display Profiles.....	32
Additional Hijri Date Configuration.....	34
Defining Phone Types.....	34
Setting Up Characteristic Types & Values.....	34
There Are Four Types Of Characteristics.....	35
Searching By Characteristic Values.....	35
Characteristic Type - Main.....	36
Characteristic Type - Characteristic Entities.....	37
Setting Up Foreign Key Reference Information.....	38
Information Description Is Dynamically Derived.....	38
Navigation Information Is Dynamically Derived.....	39
Search Options.....	39
Foreign Key Reference - Main.....	39
Defining Feature Configurations.....	40
Feature Configuration - Main.....	41
Feature Configuration - Messages.....	41
Defining Master Configurations.....	42
Defining Security & User Options.....	42
The Big Picture of Application Security.....	42
Application Security.....	42
Action Level Security.....	44
Field Level Security.....	44
Encryption and Masking.....	45
The Base Package Controls One User, One User Group, And Many Application Services.....	51
Importing Security Configuration from an External Source.....	52
The Big Picture of Row Security.....	52
Access Groups, Data Access Roles and Users.....	52
Defining Application Services.....	53
Application Service - Main.....	53
Application Service - Application Security.....	53
Defining Security Types.....	54
Security Type - Main.....	54
Defining User Groups.....	54
User Group - Main.....	54
User Group - Application Services.....	55

User Group - Users.....	56
Defining Access Groups.....	56
Defining Data Access Roles.....	56
Data Access Role - Main.....	57
Data Access Role - Access Group.....	57
Defining Users.....	57
User Interface Tools.....	57
Defining Menu Options.....	57
Menu - Main.....	58
Menu - Menu Items.....	59
The Big Picture of System Messages.....	60
Defining System Messages.....	60
The Big Picture of Portals and Zones.....	62
There Are Three Types of Portals.....	62
Common Characteristics of All Portals.....	63
Common Characteristics of Stand-Alone Portals.....	64
Custom Look and Feel Options.....	65
User Interface.....	65
UI Map Help.....	66
Setting Up Portals and Zones.....	66
Defining Zone Types.....	66
Defining Zones.....	67
Defining Context-Sensitive Zones.....	90
Defining Portals.....	90
Defining Display Icons.....	92
Defining Navigation Keys.....	92
Navigation Key Types.....	92
Navigation Key vs. Navigation Option.....	93
The Flexibility of Navigation Keys.....	93
Linking to External Locations.....	93
Overriding Navigation Keys.....	93
Maintaining Navigation Keys.....	94
Defining Navigation Options.....	95
Navigation Option - Main.....	95
Navigation Option - Tree.....	97
Database Tools.....	97
Defining Table Options.....	97
Table - Main.....	98
Table - Table Field.....	100
Table - Constraints.....	100
Table - Referred by Constraints.....	101
Defining Field Options.....	102
Field - Main.....	102
Field - Tables Using Field.....	103
Defining Maintenance Object Options.....	103
Maintenance Object - Main.....	103
Maintenance Object - Options.....	104
Maintenance Object - Algorithms.....	104
Maintenance Object - Maintenance Object Tree.....	106
Defining Valid Values.....	106
Defining Lookup Options.....	107
Defining Extendable Lookups.....	109
The Big Picture Of Audit Trails.....	110
Captured Information.....	110
How Auditing Works.....	111
The Audit Trail File.....	111
How To Enable Auditing.....	112
Audit Queries.....	113
Bundling.....	114
About Bundling.....	115
Configuring Maintenance Objects for Bundling.....	116
Working with Bundles.....	118
Revision Control.....	120
About Revision Control.....	120

Working with the Revision Control Zones.....	122
Working with the Revision Control Portal.....	124
Information Lifecycle Management	125
The Approach to Implementing Information Lifecycle Management.....	125
Enabling ILM for Supported Maintenance Objects.....	127
Ongoing ILM Tasks.....	128
Archived Foreign Keys.....	129
Configuration Tools.....	129
Business Objects.....	129
The Big Picture of Business Objects.....	129
Defining Business Objects.....	144
Advanced BO Tips and Techniques.....	150
Defining Status Reasons.....	151
Business Services.....	152
Service Program.....	152
Defining Business Services.....	153
User Interface (UI) Maps.....	157
Defining UI Maps.....	159
Ensuring Unique Element IDs for UI Maps.....	229
Maintaining Managed Content.....	230
Managed Content - Main.....	230
Managed Content - Schema.....	230
Data Areas.....	230
Defining Data Areas.....	231
Advanced Schema Topics.....	232
Schema Nodes and Attributes.....	232
UI Hint Syntax.....	250
Schema Designer.....	257
Schema Viewer.....	258
Business Event Log.....	259
Miscellaneous Topics.....	259
Module Configuration.....	259
Global Context Overview.....	260
System Data Naming Convention.....	261
Caching Overview.....	262
Expression Parser.....	263
Debug Mode.....	265
System Override Date.....	266
Advanced Search Options.....	266
To Do Lists.....	267
The Big Picture of To Do Lists.....	267
To Do Entries Reference A To Do Type.....	267
To Do Entries Reference A Role.....	267
To Do Entries Can Be Rerouted (Or Suppressed) Based On Message Number.....	268
The Priority Of A To Do Entry.....	269
Working On A To Do Entry.....	269
Launching Scripts When A To Do Is Selected.....	269
To Do Entries Have Logs.....	270
How Are To Do Entries Created?.....	270
The Lifecycle Of A To Do Entry.....	273
Linking Additional Information To A To Do Entry.....	273
Implementing Additional To Do Entry Business Rules.....	274
To Do Entries May Be Routed Out Of The System.....	274
Periodically Purging To Do Entries.....	274
Setting Up To Do Options.....	274
Installation Options.....	275
Messages.....	275
Feature Configuration.....	276
Defining To Do Roles.....	276
Defining To Do Types.....	277
List of System To Do Types.....	281
Implementing The To Do Entries.....	281
Background Processes.....	281
Understanding Background Processes.....	281

Background Processing Overview.....	282
Parallel Background Processes.....	283
Parameters Supplied To Background Processes.....	284
Processing Errors.....	287
Post-Processing Logic.....	287
Timed Batch Processes.....	287
Monitor Background Processes.....	288
Plug-in Driven Background Processes.....	289
How to Re-extract Information.....	291
How to Submit Batch Jobs.....	291
How to Track Batch Jobs.....	291
How to Restart Failed Jobs and Processes.....	291
Assessing Level of Service.....	292
System Background Processes.....	292
Defining Batch Controls.....	292
Batch Control - Algorithms.....	294
On-Line Batch Submission.....	295
Batch Submission Creates a Batch Run.....	295
Jobs Submitted in the Background.....	296
Email Notification.....	296
Running Multi-Threaded Processes.....	296
Batch Jobs May End in Error.....	296
Submitting Jobs in the Future.....	297
Lifecycle of a Batch Job Submission.....	297
Batch Job Submission - Main.....	297
Tracking Batch Processes.....	299
Batch Run Tree - Main.....	299
Batch Run Tree - Run Control.....	300
Service Health Check.....	300
The Big Picture of Requests.....	301
Request Type Defines Parameters.....	301
Previewing and Submitting a Request	302
To Do Summary Email.....	302
Exploring Request Data Relationships.....	302
Defining a New Request.....	302
Setting Up Request Types.....	303
Maintaining Requests.....	303
Defining Algorithms.....	304
The Big Picture Of Algorithms.....	304
Algorithm Type Versus Algorithm.....	304
How To Add A New Algorithm.....	304
Minimizing The Impact Of Future Upgrades.....	305
Setting Up Algorithm Types.....	305
List of Algorithm Types.....	306
Setting Up Algorithms.....	307
Defining Script Options.....	307
The Big Picture Of Scripts.....	307
Scripts Are Business Process-Oriented.....	308
A Script Is Composed Of Steps.....	308
Designing And Developing Scripts.....	308
A Script May Declare Data Areas.....	309
Designing Generic Scripts.....	309
Securing Script Execution.....	309
The Big Picture Of BPA Scripts.....	309
How To Invoke Scripts.....	310
Developing and Debugging Your BPA Scripts.....	310
Launching A Script From A Menu.....	310
Launching A Script When Starting The System.....	311
Executing A Script When A To Do Entry Is Selected.....	311
The Big Picture Of Script Eligibility Rules.....	312
The Big Picture Of Server-Based Scripts.....	316
Plug-In Scripts.....	316
Service Scripts.....	317
Debugging Server-Based Scripts.....	318

Maintaining Scripts.....	318
Script - Main.....	318
Script - Step.....	319
Script - Data Area.....	356
Script - Schema.....	356
Script - Eligibility.....	356
Merging Scripts.....	358
Script Merge.....	359
Maintaining Functions.....	362
Function - Main.....	362
Function - Send Fields.....	363
Function - Receive Fields.....	364
Attachments.....	364
Attachment Overview.....	364
Configuring Your System for Attachments.....	365
Maintaining Attachments.....	366
Adding Attachments.....	367
Application Viewer.....	367
Application Viewer Toolbar.....	367
Data Dictionary Button.....	367
Physical and Logical Buttons.....	368
Collapse Button.....	368
Attributes and Schema Button.....	368
Maintenance Object Button.....	368
Algorithm Button.....	369
Batch Control Button.....	369
To Do Type Button.....	369
Description and Code Buttons.....	369
Service XML Button.....	369
Select Service Button.....	370
Java Docs Button.....	370
Classic Button.....	370
Preferences Button.....	370
Help Button.....	370
About Button.....	370
Slider Icon.....	371
Data Dictionary.....	371
Using the Data Dictionary List Panel.....	371
Using the Data Dictionary Detail Panel.....	372
Maintenance Object Viewer.....	373
Using the Maintenance Object List Panel.....	373
Using the Maintenance Object Detail Panel.....	373
Algorithm Viewer.....	374
Using the Algorithm Viewer List Panel.....	374
Using the Algorithm Plug-In Spot Detail Panel.....	374
Using the Algorithm Type Detail Panel.....	374
Using the Algorithm Detail Panel.....	374
Batch Control Viewer.....	374
Using the Batch Control Viewer List Panel.....	375
Using the Batch Control Detail Panel.....	375
To Do Type Viewer.....	375
Using the To Do Type Viewer List Panel.....	375
Using the To Do Type Detail Panel.....	375
Service XML Viewer.....	376
Using the Service XML Viewer Overview Panel.....	376
Using the Service XML Viewer Detail Panel.....	376
Java Docs Viewer.....	376
Using the Java Docs Viewer List Panel.....	377
Using the Java Package Detail Panel.....	377
Using the Java Interface / Class Detail Panel.....	377
Application Viewer Preferences.....	377
Application Viewer Stand-Alone Operation.....	377
Stand-Alone Configuration Options.....	378
Example Application Viewer Configuration.....	378

Application Viewer Generation.....	379
Defining and Designing Reports.....	379
The Big Picture Of Reports.....	379
Integration with BI Publisher and Business Objects Enterprise.....	380
How To Request Reports.....	381
Viewing Reports.....	382
Configuring The System To Enable Reports.....	382
Configuring BI Publisher Reports.....	382
Configuring Business Objects Enterprise Reports.....	383
Defining Reporting Options.....	383
Defining Report Definitions.....	384
Sample Reports Supplied with the Product.....	385
How to Use a Sample Report Provided with the System.....	385
Subreports Used with Crystal Reports.....	386
How To Define A New Report.....	387
Use a Sample Report as a Starting Point.....	387
Publishing Reports in BI Publisher.....	387
Publishing Reports in Business Objects Enterprise.....	388
Designing Your Report Definition.....	389
External Messages.....	391
Incoming Messages.....	391
Inbound Web Services.....	391
Guaranteed Delivery.....	396
Outgoing Messages.....	396
Outbound Messages.....	396
Web Service Adapters.....	411
Sending Email.....	413
JMS Message Browser.....	414
XML Application Integration.....	414
The Big Picture of XAI.....	414
Designing Your XAI Environment.....	430
Schema Editor.....	439
Setting Up Your XAI Environment.....	446
Running Your XAI Environment.....	454
Maintaining Your XAI Environment.....	459
Integrations.....	463
LDAP Integration.....	463
LDAP Integration Overview.....	463
Configuring LDAP Integration	464
Oracle Identity Manager Integration.....	467
Batch Scheduler Integration.....	469
Data Synchronization.....	469
About Data Synchronization.....	469
Maintaining Sync Requests.....	471
Analytics Integration.....	471
About Analytics Integration.....	471
Maintaining Bucket Configurations.....	472
ETL Mapping Control.....	473
Business Flags.....	473
About Business Flags.....	473
Setting Up Business Flag Configuration.....	477
Maintaining Business Flags.....	478
Configuration Migration Assistant (CMA).....	478
Understanding CMA.....	478
The CMA Process Flow.....	479
Migration Assumptions, Restrictions and Recommendations.....	480
CMA Configuration.....	483
Master Configuration - Migration Assistant.....	483
Migration Plans.....	483
Defining a Migration Request.....	486
Wholesale and Piecemeal Migrations.....	486
Identifying Tables to Exclude From Migrations.....	488
Configuring Custom Objects for Migration.....	488
The CMA Execution Process.....	489

Exporting a Migration.....	490
Importing and Applying a Migration.....	492
Running Batch Jobs.....	507
CMA Reference.....	507
Framework-Provided Migration Configuration.....	508
Configuring Facts.....	510
Fact Is A Generic Entity.....	510
Fact's Business Object Controls Everything.....	510
Fact Supports A Log.....	511
Customer Care and Billing Administrative User Guide.....	511
Defining General Options.....	511
Defining Installation Options.....	511
Installation Options - Main.....	511
Installation Options - Person.....	513
Installation Options - Account.....	513
Installation Options - Billing.....	514
Installation Options - C&C.....	515
Installation Options - Financial Transaction.....	516
Installation Options - Algorithms.....	516
Defining Customer Languages.....	523
Defining Accounting Calendars.....	523
Defining General Ledger Divisions.....	524
Defining Banks & Bank Accounts.....	524
Bank - Main.....	525
Bank - Bank Account.....	525
Defining Issuing Centers.....	525
Actions.....	526
Issuing Center List.....	526
Issuing Center.....	526
Issuing Center Log.....	526
Setting Up Service Types.....	526
Service Type - Main.....	527
Service Type - Level 1.....	527
Service Type - Level 2.....	527
Service Type - Level 3.....	527
Defining Service Tasks Options.....	528
About Service Tasks.....	528
About Service Task Types.....	528
Defining Service Task Types.....	528
Base Package Service Task Types.....	529
Defining Financial Transaction Options.....	531
The Financial Big Picture.....	531
Bills, Payments & Adjustments.....	532
Bill Details.....	532
Payment Details.....	533
Adjustment Details.....	534
Current Amount versus Payoff Amount.....	535
Financial Transactions Created Between Bills.....	538
Financial Transactions And Aged Debt.....	540
Preventing SA Balances And The GL From Being Impacted Until Bill Completion.....	540
Forcing The Freeze Date To Be Used As The Accounting Date.....	542
How Late Payment Charges Get Calculated.....	543
Service Agreement Type Controls Everything.....	544
Setting Up CIS Divisions.....	545
Setting Up Revenue Classes.....	546
Setting Up Distribution Codes.....	547
Setting Up Billable Charge Templates.....	548
Designing and Defining Bill Segment Types.....	550
Designing and Defining Deposit Classes.....	555
Setting Up Payment Segment Types.....	560
Designing And Defining Adjustment Types.....	561
Designing and Defining Budget Plans.....	574
The Financial Impact Of Budget Plans.....	574
What Do Budget Plans Do?.....	574

Designing Your Budget Plans.....	574
Setting Up Budget Plans.....	575
Tender Management.....	576
Setting Up Tender Types.....	577
Setting Up Tender Sources.....	578
Automatic Payment Options.....	580
Setting Up Auto-Pay Route Types.....	580
Setting Up Auto-Pay Source Codes.....	581
SEPA Payments.....	582
Payment Advices.....	584
What Is A Payment Advice?.....	584
Payment Advice vs. Direct Debit.....	585
Setting Up The System To Enable Payment Advices.....	585
Credit Card Payments.....	585
Configuring the System for Tender Authorization.....	585
Non CIS Payments.....	587
Setting Up Payment Templates.....	587
Alternate Currency Payments.....	588
What Is An Alternate Currency Payment?.....	588
Configuring the System for Alternate Currency Payments.....	588
Payment Event Distribution.....	589
Making Payments Using Distribution Rules.....	589
Setting Up The System To Use Distribution Rules.....	590
Cancel Reasons.....	591
Setting Up Bill (Segment) Cancellation Reasons.....	591
Setting Up Payment Cancellation Reasons.....	592
Setting Up Adjustment Cancellation Reasons.....	593
Miscellaneous Financial Controls.....	593
A/P Check Request.....	593
Billable Charge Line Type.....	593
Payables Cash Accounting.....	594
Accrual versus Cash Accounting Example.....	594
Distribution Code Controls Cash Accounting For A GL Account.....	595
Bill Segments and Cash Accounting.....	596
Payment Segments and Cash Accounting.....	596
Write Down Adjustment.....	605
Write-Offs.....	606
Deferred Accrual Accounting.....	606
Deferred Accrual Accounting Examples.....	607
Payment Cancellations and Deferred Accrual Accounting.....	609
Open Item Accounting.....	610
Open-Item Versus Balance-Forward Accounting.....	610
Accounting Method Defined On Your Customer Classes.....	611
Match Events.....	611
Disputing Items.....	616
Pay Plans.....	618
Over Payments.....	618
Setting Up The System To Enable Open Item Accounting.....	619
Setting Up Match Types.....	621
Setting Up Match Event Cancellation Reasons.....	622
Fund Accounting.....	622
Fund Accounting Overview.....	622
Accounting Method Is Defined On The Installation Options.....	626
Fund Controls Fund-Balancing Entries.....	626
Building Fund-Balancing GL Details.....	626
Setting Up The System To Enable Fund Accounting.....	628
United Kingdom VAT and CCL.....	629
UK VAT Overview.....	629
UK CCL Overview.....	629
UK VAT and CCL Bill Examples.....	630
Billing and UK VAT.....	631
Excess Credits and UK VAT.....	632
Setting Up The System For UK VAT and CCL.....	633
Bill Taxation Threshold.....	637

Taxation Threshold Examples.....	637
Billing and Taxation Thresholds.....	639
Setting Up The System For Bill Taxation Thresholds.....	640
Other Financial Transaction Topics.....	642
The Source Of GL Accounts On Financial Transactions.....	642
Defining Customer Options.....	643
Customer Overview.....	644
A Simple Example Of Two Customers.....	644
Persons.....	644
Accounts.....	645
Service Agreements.....	646
Setting Up Person Options.....	647
Setting Up Identifier Types.....	647
Setting Up Person Relationship Types.....	647
Setting Up Contact Routings.....	648
Setting Up Person Contact Status.....	648
Setting Up Person Contact Type Algorithms.....	648
Setting Up Person Contact Types.....	649
Person Contact Type Portal.....	649
Choosing to Use Person Contact or Person Phone.....	650
Person Contact Status Can Be Controlled By A Process.....	650
Maintaining Person Phone via Person Contact.....	652
Setting Up Statement Construct Options.....	653
Setting Up Statement Route Types.....	653
Setting Up Account Options.....	653
Setting Up Account Management Groups.....	654
Setting Up Account Relationship Codes.....	654
Setting Up Alert Types.....	654
Setting Up Bill Messages.....	655
Setting Up Bill Route Types.....	655
Setting Up Bill Cycles.....	656
Setting Up Customer Classes.....	656
Setting Up Collection Classes.....	663
Setting Up Customer Information Options.....	663
Setting Up Customer Contact Options.....	663
Setting Up Customer Contact Classes.....	664
Setting Up Customer Contact Types.....	664
Setting Up Notification Preference Options.....	664
Notification Preferences Overview.....	665
Push vs. Subscription Notification Types.....	665
Parent vs. Individual Push Notification Types.....	665
Delivery Type.....	666
Enabling Opt-In for a Delivery Type.....	666
Some Notification Types Use Service Task.....	666
Designing Notification Types.....	667
Setting Up Notification Type Algorithms.....	668
Setting Up Delivery Types.....	669
Setting Up Notification Types.....	669
Notification Type Portal.....	669
Setting Up the Outbound Message Type.....	670
Setting Up the Message Sender.....	670
Setting Up the External System and Configure the Messages.....	670
Setting Up Notification Preferences Master Configuration.....	670
Other Edge Application Notification.....	670
Setting Up Service Agreement Options.....	670
Setting Up Standard Industry Codes (SIC).....	671
Setting Up Tax Exempt Types.....	671
Setting Up Contract Quantity Types.....	671
SA Type Controls Everything.....	671
Financial Controls.....	672
Setting Up Order Options.....	672
Setting Up Column References.....	672
Setting Up Order Cancellation Reasons.....	673
Setting Up Order Hold Reasons.....	673

Setting Up Order Feature Configurations.....	673
Setting Up Program Management Options.....	674
Designing Initiative Eligibility Criteria.....	674
Designing Related Object Criteria.....	675
Setting Up Lead Event Types.....	676
Overriding Initiative Sign Up Options.....	677
Program Management Master Configuration.....	678
Defining Field Order Options.....	678
An Example Of A Field Order.....	678
An Example Of The Entities Involved In Field Order Dispatch.....	679
Setting Up Representatives.....	680
Setting Up Operations Areas.....	680
Setting Up Dispatch Groups.....	681
Dispatch Group - Main.....	681
Dispatch Group - Field Activity Type Review.....	682
Dispatch Group - Algorithm.....	682
Defining Disconnect Locations.....	684
Designing Your Field Activity Profiles and Types.....	684
How Does A Field Activity Type Profile Get Used?.....	685
Designing Field Activity Type Profiles.....	685
Designing Field Activity Types.....	691
Designing Field Service Classifications.....	696
Designing Who Does Your Field Activities.....	696
Setting Up Field Service Classification.....	698
Setting Up Field Activity Types.....	698
Field Activity Type - Main.....	699
Field Activity Type - FA Characteristics.....	701
Field Activity Type - FA Completion Control.....	701
Field Activity Type - SP Type Review.....	702
Setting Up Field Service Control.....	702
Setting Up Field Activity Type Profiles.....	703
Field Activity Type Profile - Main.....	703
Field Activity Type Profile - Template.....	703
Field Activities Initiated To Start Service.....	704
Field Activities Initiated To Stop Service.....	704
Field Activities Initiated For Back-to-Back Service.....	704
Field Activities Initiated To Cut Service Due To Non-Payment.....	705
Field Activities Initiated To Place A Disconnect Warning At A Service Point.....	705
Field Activities Initiated To Reconnect Service At A Service Point.....	705
Field Activities Initiated To Reread A Meter At A Service Point.....	706
Defining A Profile's Valid Field Activity Types.....	706
Setting Up Fieldwork Cancellation Reasons.....	706
Fieldwork Reschedule Reason.....	707
Setting Up Field Activity Remarks.....	707
Setting Up Outage Call Types.....	708
Outage Call Type List.....	708
Outage Call Type.....	708
Defining Credit & Collections Options.....	709
The Big Picture Of Credit & Collections.....	709
Collection Criteria vs. Severance Criteria vs. Write Off Criteria.....	709
The C&C Monitors.....	712
The Big Picture Of Collection Processes.....	714
The Big Picture Of Collection Events.....	721
The Big Picture Of Severance Process Cancellation.....	724
The Big Picture Of Severance Events.....	726
The Big Picture Of Write Off Processing.....	730
The Big Picture Of Write-off Events.....	737
Calendar vs. Work Days.....	740
The Big Picture Of Payment Arrangements and Pay Plans.....	741
Creating Collection, Severance & Write-Off Procedures.....	751
Designing Your Collection Procedures.....	752
Setting Up Collection Procedures.....	758
Designing Your Severance Procedures.....	764
Designing Your Reconnection Procedures.....	766

Setting Up Severance Procedures.....	768
Designing Your Write-Off Procedures.....	770
Setting Up Write-Off Procedures.....	773
Setting Up Feature Configuration.....	778
How To.....	779
How To Nominate A Single Service To Sever (Rather Than Sever Everything That's In Arrears).....	779
Defining Meter & Item Options.....	781
The Big Picture of Meters, Items and Equipment.....	781
The Structure Of A Meter.....	781
Items Are Used For Other Devices Associated With A Customer's Service.....	782
Setting Up Meter Options.....	784
Setting Up Meter Configuration Types.....	784
Setting Up Meter Types.....	785
Setting Up Manufacturers & Their Models.....	786
Setting Up Meter ID Types.....	787
Setting Up Read Out Types.....	787
Setting Up Protocol Codes.....	787
Setting Up Unit Of Measure Codes.....	787
Setting Up Time-Of-Use Codes.....	788
Setting Up TOU Groups.....	789
Setting Up Retirement Reasons.....	789
Setting Up Metered Service Point Options.....	790
Defining Meter Location Codes.....	790
Setting Up Service Cycles And Routes.....	790
Setting Up Metered Premise Options.....	790
Setting Up Meter Read Instructions.....	790
Setting Up Meter Read Warnings.....	790
Setting Up Consumption Estimation Parameters.....	791
Estimating Consumption.....	791
Setting Up Trend Areas.....	797
Setting Up Trend Classes.....	798
Setting Up High / Low Factors.....	798
Setting Up Trends.....	799
Setting Up Meter Read Options.....	799
Setting Up Meter Reader Remarks.....	799
Setting Up Meter Read Sources.....	800
Setting Up Items.....	801
Setting Up Item Types.....	801
Setting Up Manufacturers and Models.....	803
The Big Picture Of Device Testing.....	803
The Level of Complexity Depends On What You Test and Your Record Keeping Requirements.....	803
A Device Test Records Test Results.....	804
Field Activities And Device Testing.....	805
Device Test Validation.....	805
Setting Up Device Test Options.....	805
Setting Up Component Test Types.....	806
Setting Up Device Test Types.....	806
Usage Administration.....	807
The Big Picture Of Usage Requests.....	808
The Big Picture of Time of Use Mapping and Pricing.....	812
Defining Premise & Service Point Options.....	819
An Illustration Of A Premise.....	819
Setting Up Premise Options.....	821
Defining Premise Types.....	821
Implementing Address Validation	821
Setting Up Generic Service Point Options.....	822
Facility Levels.....	822
Setting Up Service Point Types.....	822
Setting Up Premise & Service Point Postal Defaults.....	825
Postal Defaults - Main.....	826
Postal Defaults - Service Default.....	826
Designing SP Types.....	827
Service Segmentation.....	828
Device Segmentation.....	828

Meter Read Estimation Trend Class Segmentation.....	828
Field Activity Type Profile Segmentation.....	829
SA Type Segmentation.....	829
Meter Type Segmentation.....	830
Item Type Segmentation.....	830
Defining Bill & Meter Read Cycles.....	830
Defining Bill & Service Cycles.....	830
The Big Picture Of Bill Cycles, Service Cycles and Bill Periods.....	831
Designing Cycles for Waste Collection Services.....	834
Setting Up Bill Cycles.....	839
Setting Up Service Route Types.....	840
Setting Up Service Cycles And Routes.....	840
Setting Up Service Cycle Schedules.....	841
Setting Up Bill Periods.....	842
Defining Statement Cycles.....	842
The Big Picture Of Statement Cycles.....	842
Setting Up Statement Cycles.....	843
Defining Service Agreement Type (SA Types).....	844
Designing SA Types For Service Agreements With Service Points.....	844
CIS Division Segmentation.....	844
Service Segmentation.....	845
Receivable Segmentation.....	845
Revenue Segmentation.....	846
Rate Segmentation.....	847
Service Point (SP) Type Segmentation.....	848
Company Usage Segmentation.....	848
Debt Class Segmentation.....	849
Budget Billing Segmentation.....	850
Designing SA Types For SAs Without Service Points.....	852
Overpayment Segmentation.....	852
Write Off Segmentation.....	853
Connection Charge Segmentation.....	854
Charitable Contribution Segmentation.....	854
Payment Arrangement Segmentation.....	855
Merchandise Segmentation - Installment Billing.....	856
Deposit Segmentation - Installment Billing.....	857
Billable Charge Segmentation.....	858
Over/Under Cash Drawer Segmentation.....	859
Payment Upload Error Segmentation.....	860
CIAC Segmentation.....	860
Loan Segmentation.....	861
Non-billed Budget Segmentation.....	861
Designing SA Type For Other Segmentations.....	862
Cash Distribution Segmentation.....	862
Adjustment Profile Segmentation.....	862
Late Payment Charge Segmentation.....	862
Debt Classification Segmentation.....	863
Allow Estimates Segmentation.....	863
Severance Criteria Segmentation.....	863
Deposit Class Segmentation.....	864
Sub SA Types.....	864
Financial Settlement SA Types.....	864
Interval Billing SA Types.....	864
Usage Request SA Types.....	865
Initial Consumption Period Considerations.....	865
Processing Sequence Considerations.....	867
Designing Prepayment Billing Options.....	867
SA Types And The Financial Design.....	869
Setting Up SA Types.....	870
SA Type - Main Information.....	870
SA Type - Detail.....	873
SA Type - Billing.....	875
SA Type - Rate.....	879
SA Type - SP Type.....	880

SA Type - Adjustment Profiles.....	881
SA Type - C&C.....	881
SA Type - Billable Charge Template.....	882
SA Type - Characteristics.....	883
SA Type - Algorithms.....	883
SA Type - Billable Charge Overrides.....	891
SA Type - Contract Option Type.....	891
SA Type - Interval Info.....	892
SA Type - NBB Recommendation Rule.....	893
Setting Up Start Options.....	893
Start Option Merge.....	901
Defining SA Relationship Options.....	905
The Big Picture of SA Relationships and Service Providers.....	905
Persons and Service Providers.....	906
Service Providers Are Linked To Service Agreements.....	906
Service Providers May Change Over Time.....	908
How To Set Up SA Relationships On A Customer's Service Agreement.....	908
When Your Company Is A Service Provider.....	909
Service Providers Have To Communicate About Customers.....	910
Relationships Between Service Providers.....	911
A Service Agreement Can Have Many Types Of Relationships.....	911
Billing Relationships.....	913
Consumption Relationships.....	930
Deposits Issues.....	933
Credit and Collection Issues.....	934
An Object-Oriented Perspective Of Service Providers.....	936
How Do You Communicate With Service Providers?.....	937
Off Cycle Switch.....	937
Designing Your SA Relationship Types and Service Providers.....	938
Designing SA Relationship Types.....	938
Designing Service Providers.....	939
Designing Your SA Types And Start Options For Sub SAs.....	945
Reference Send Consumption Algorithm On Master SA Types.....	947
Reference TBFU Algorithm On Master SA Types.....	947
Designing SA Types For Service Provider Financial Settlements.....	948
Setting Up SA Relationship Information.....	948
Setting Up SA Relationship Types.....	948
Setting Up Service Providers.....	949
Setting Up SA Types and Start Options For Sub SAs.....	951
Setting Up SA Types For Financial Settlements.....	951
Update Master SA Types With FT Freeze and Bill Completion Algorithms.....	951
Setting Up SA Relationships For SA Types.....	951
Negotiated Terms.....	952
Examples Of Special Discounts.....	953
Setting Up The System To Enable Negotiated Terms.....	954
Defining Service Credit Options.....	956
The Big Picture Of Service Credits.....	956
Service Credit Membership.....	956
Designing Your Service Credit Options.....	959
Designing Your Membership Types.....	959
Designing Your Service Credit Event Types.....	962
How Are Service Credit Events Created?.....	965
Setting Up Service Credit Options.....	966
Setting Up Credit Units.....	966
Setting Up SC Membership Inactive Reasons.....	966
Setting Up Service Credit Membership Types.....	967
Setting Up Service Credit Event Types.....	969
Service Credit Examples.....	969
Defining Control Tables for a Refundable Fee.....	970
Defining Control Tables for a Nonrefundable Fee.....	971
Defining an SA Type for Miscellaneous Transactions.....	971
Using Campaigns/Packages to Set Up Membership.....	972
Defining Another Person for Your Account.....	974
Service Credits Earned When Starting Service.....	977

Service Credits Earned Through Billing.....	977
Service Credits Redeemed Through Billing.....	978
Designing Your Rate Options for Capital Credits.....	978
Partial Retirement.....	981
Interface Membership Information to a Third Party.....	981
Defining Loan Options.....	982
The Terms Of A Loan Are Stored On A Service Agreement.....	982
Payoff Balance and Current Balance for Loans.....	983
Booking The Principal Amount Using An Adjustment.....	983
Loan Amortization Schedule Calculation.....	984
Billing For Loans And Interest Calculation.....	984
Paying What Is Owed.....	986
Overpayments On Loans.....	987
Adjusting Loan Amounts.....	989
Writing Off Loans.....	990
Distribution Codes for Loans.....	992
Setting Up The System To Enable Loans.....	992
Distribution Code.....	992
Adjustment Types.....	992
Adjustment Type Profile.....	993
Algorithms.....	993
Bill Factor.....	993
Customer Class.....	994
Bill Segment Type.....	994
Frequency.....	994
Bill Period.....	994
Bill Cycle Schedule.....	994
Collection, Severance and Write Off Processes.....	994
SA Type.....	994
Start Options.....	996
Defining Non-billed Budget Options.....	997
What Is A Non-billed Budget?.....	997
The Financial Impact Of Non-billed Budgets.....	997
Current Amount For SAs Covered By A Non-billed Budget.....	997
Scheduled And Actual Payments On The Non-billed Budget.....	998
Overpayments for Non-billed Budgets.....	999
Underpayments For Non-billed Budgets.....	1000
Billing For SAs Covered By The Non-billed Budget.....	1001
Distributing Non-billed Budget Credit.....	1002
Stopping an SA Covered By a Non-billed Budget.....	1003
Financial Transactions For Unmonitored Non-billed Budgets.....	1004
Distributing Payments for Unmonitored Non-billed Budgets.....	1005
Transferring Credit from Unmonitored Non-billed Budgets.....	1005
Designing Non-billed Budgets.....	1006
Making SAs Eligible For Non-billed Budgets.....	1006
Designing Recommendation Rules.....	1006
Non-billed Budget Recommendation Rule.....	1012
Setting Up The System To Enable Non-billed Budgets.....	1013
NBB Distribution Codes.....	1013
NBB Adjustment Types.....	1013
NBB Adjustment Type Profiles.....	1014
NBB Characteristic Types.....	1014
NBB Customer Contact Class And Types.....	1015
NBB Algorithms.....	1015
NBB Debt Class And Collection Process.....	1016
SA Types for SAs Covered by NBBs.....	1016
NBB Recommendation Rules.....	1016
Non-billed Budget SA Types.....	1017
NBB Background Processes.....	1018
Defining Quotation Options.....	1019
Setting Up SA Types For Quotes.....	1019
Enabling The Automatic Generation Of Billing Scenarios.....	1019
Enabling The Generation Of Simulated Bill Segments.....	1019
Enabling The Creation Of A Real SA When A Quote Detail Is Accepted.....	1019

Setting Up Quote Route Types.....	1020
Setting Up Terms and Conditions.....	1020
Setting Up Decline Reasons.....	1021
Setting Up Customer Classes For Quotes.....	1021
Defining Case Management Options.....	1021
The Big Picture Of Cases.....	1022
Case Type Controls Everything.....	1022
Scripts and Cases.....	1029
To Do's and Cases.....	1029
Examples of Case Types.....	1032
Setting Up Case Management Options.....	1041
Installation Options.....	1041
Setting Up Application Services.....	1042
Setting Up Scripts.....	1042
Setting Up To Do Types.....	1042
Setting Up Characteristic Types.....	1043
Setting Up Case Types.....	1043
Workflow and Notification Options.....	1047
The Big Picture Of Workflow Processing.....	1048
The Big Picture Of Workflow Events.....	1048
How Are Workflow Events Created?.....	1048
Executing Workflow Events On Their Trigger Date.....	1048
Workflow Event Lifecycle.....	1049
Workflow Event Dependencies & Trigger Date.....	1050
Workflow Events May Be In Error.....	1051
Some Workflow Events May Fail.....	1051
Errors versus Failure.....	1052
Waiting Events And Their Waiting Process.....	1052
How Are Workflow Events Canceled?.....	1053
Workflow Processes Can Have Multiple Branches.....	1053
The Big Picture of Notification Processing.....	1056
Uploading Notifications Into The System.....	1056
What Type Of Workflow Process Is Created?.....	1057
How Are Notifications Sent Out Of The System?.....	1057
System Conditions May Trigger Notification and Workflow.....	1066
The Lifecycle of Notification Download Staging.....	1068
Creating Notification and Workflow Procedures.....	1068
Designing Notification Upload & Workflow Procedures.....	1069
Designing Notification Downloads.....	1076
Setting Up Notification and Workflow Procedures.....	1082
Defining Umbrella Agreement Options.....	1088
The Big Picture Of Umbrella Agreements.....	1088
Renewable Umbrella Agreements.....	1089
Terms Of Service Covered Entities.....	1089
Overriding Rate Terms.....	1089
Setting Up Umbrella Agreement Options.....	1091
Configure SA Types for Umbrella Agreements.....	1091
Configure Bill Factors for Umbrella Agreements.....	1092
Configure A Campaign To Link An SA To A Terms Of Service.....	1092
Setting Up Umbrella Agreement Types.....	1092
Setting Up Terms Of Service Types.....	1094
Setting Up TOS Cancel Reasons.....	1094
Umbrella Agreements Configuration Examples.....	1095
Linking A Service Agreement To A TOS Through Orders.....	1095
Reports Addendum.....	1096
Description of Sample Reports.....	1096
Active Severance Processes - CI_ACSVPR.....	1096
Bill Print in BI Publisher - CI_BILLPR.....	1096
Billed Revenues by Rate - CI_BILREV.....	1104
Case Statistics By Case Type - CI_CSESTS.....	1105
Case Statistics for a Given Status - CI_CSESGS.....	1105
Collection Summary - CI_CLLSUM.....	1106
Customer Contacts by Type - CI_CUSTCN.....	1106
Customers with Life-Support / Sensitive-Load - CI_PMLSSL.....	1107

Field Order Print in BI Publisher- CI_FOPRNT.....	1107
GL Accounting Summary - CI_GLACSM.....	1115
Letter Print - BI Publisher Sample Welcome Letters - CI_LTRGN_ENG REPORT.....	1115
Meter Reads Performance - CI_MTREAD.....	1116
Open Cases By Type - CI_CSEOPN.....	1117
Payments Balance - CI_PMTBAL.....	1117
Receivables Aging - CI_RCVAGA.....	1118
Tax Payables Analysis - CI_TXPYBL.....	1118
To Do Entries - CI_TDENTR.....	1119
Umbrella Agreement Summary - CI_UASUMM.....	1120
Unbilled Revenues - CI_UBLREV.....	1121
Vacant Premises with Consumption - CI_VACANT.....	1122
Defining Overdue Processing Options.....	1122
Case Study - Collecting On Overdue Bills.....	1123
Monitoring Overdue Bills.....	1123
Cutting Service Agreements.....	1125
How Does The Overdue Monitor Work?.....	1127
Different Overdue Rules For Different Customers.....	1127
Overdue Rules Are Embodied In Algorithms.....	1128
When Is An Account Monitored?.....	1128
Collection Class Defines If And How Accounts Are Monitored.....	1128
The Big Picture Of Overdue Processes.....	1129
How Are Overdue Processes Created?.....	1129
The Components Of An Overdue Process.....	1129
Experimenting With Alternative Overdue Process Templates.....	1130
Overdue Process Information Is Overridable.....	1131
Original and Unpaid Amounts.....	1131
Overdue Processes Are Highlighted Elsewhere.....	1132
How Are Overdue Processes Cancelled?.....	1132
Overdue Processes Are Created From Templates.....	1133
The Big Picture Of Overdue Events.....	1133
The Big Picture Of Cut Processes.....	1136
Cut Events Are Like Overdue Events.....	1138
Write Offs Are Implemented Using Overdue Events.....	1139
Calendar vs. Work Days.....	1139
Bill-Oriented Collection - Advanced Topics.....	1139
Miscellaneous Bill-Oriented Collection Topics.....	1140
Writing Off Bills.....	1141
Bill-Oriented Payment Arrangements.....	1144
Creating Overdue and Cut Procedures.....	1147
Designing Your Overdue Procedures.....	1147
Set Up Tasks.....	1153
Setting Up Overdue Processing.....	1157
Defining Interval Billing Options.....	1164
Interval Billing Table Setup Sequence.....	1164
The Big Picture of Interval Billing.....	1166
Interval Pricing.....	1167
Raw Data Collection and Aggregation.....	1169
Time of Use Billing.....	1169
Designing Interval Billing Options.....	1175
Designing Your Interval Billing Rate Options.....	1176
Designing Your Interval Billing Controls.....	1177
Designing Your Raw Data Options.....	1179
Designing Your Time of Use Options.....	1181
Designing Your Contract Options.....	1188
Designing Your SA Interval Billing Options.....	1189
Setting Up Interval Billing Options.....	1192
Setting Up Interval Billing Control Tables.....	1192
Setting Up Channel Control Tables.....	1195
Setting Up Time of Use Billing Control Tables.....	1196
Setting Up Contract Option Control Tables.....	1201
Defining Prepaid Metering Options.....	1203
The Big Picture of Prepaid Metering.....	1203
How Does Prepaid Metering Work?.....	1203

Debit Meter vs. Credit Meter.....	1204
Prepaid Metering Transactions Result in Adjustments.....	1204
Interfacing Prepaid Transactions.....	1204
Prepaid Transactions Can Go Into Suspense.....	1204
Setting Up The System To Enable Prepaid Metering.....	1205
Meter Types.....	1205
Characteristic Types.....	1205
Algorithms.....	1206
Rate.....	1207
Adjustment Types.....	1208
Adjustment Type Profiles.....	1208
Service Agreement Types.....	1208
Bill Cycle.....	1208
Vendor Information.....	1208
Conservation Programs.....	1208
The Big Picture of Conservation Programs.....	1209
Conservation Program Maintenance Object.....	1209
Conservation Program Rebate Definition Maintenance Object.....	1209
Rebate Claim Maintenance Object.....	1210
Rebate Claim Line Maintenance Object.....	1210
GL Accounting Example.....	1211
Setting Up Conservation Programs.....	1211
Conservation Program List Zone.....	1211
Conservation Program Zone.....	1212
Rebate Definition Zone.....	1214
Rebate Claim Statistics Zone.....	1215
Configuration.....	1216
Configuring Zones.....	1216
Configuring Timeline Zones.....	1216
Configuration Migration Assistant (CMA) Addendum.....	1217
Base Package Migration Plans.....	1217
Base Package Migration Request.....	1218
Wholesale and Piecemeal Migrations.....	1218
Processing Notes for Specific Objects.....	1219
To Do Lists Addendum.....	1219
Assigning A To Do Role.....	1220
System To Do Types.....	1220
Background Processes Addendum.....	1220
Batch Process Dependencies.....	1221
How To Set Up A New Extract Processes.....	1225
The Big Picture of Sample & Submit.....	1226
Defining Batch Schedule Options.....	1229
The Big Picture of Scheduling Batch Jobs.....	1230
Setting Up The Batch Scheduler.....	1234
Maintaining Job Stream Creation Schedules.....	1237
If You Work In A Non-English Language.....	1238
Security Addendum.....	1238
Implementing Account Security.....	1238
Masking Sensitive Data.....	1246
Encrypting Sensitive Data.....	1246
Defining Converted COBOL Program Options.....	1248
Converted COBOL Program - Main.....	1248
Integration.....	1249
CTI-IVR Integration.....	1249
Launching The System From an External Application.....	1249
Receiving the Next Caller in the Queue.....	1249
Initiating an External Call.....	1250
Analytics Integration Overview.....	1251
Analytics Configuration.....	1251
Self-Service Integration.....	1252
About Self-Service Tasks.....	1252
Setting Up Self-Service Task Configuration.....	1253
Maintaining Self-Service Tasks.....	1253
Defining DataConnect Options.....	1254

DataConnect Data Extracts.....	1254
Master Data Extracts.....	1254
Billing Data Extracts.....	1258
Extract Flat File Formats.....	1260
External Fieldwork System Integration.....	1264
The Big Picture of External System Integration.....	1264
Integration Through XAI.....	1267
Dispatching Field Activities.....	1267
Incoming Messages from the External System.....	1271
Booking Appointments Via An External System.....	1276
Validating Meter / Item Installations.....	1281
Setting Up The System To Enable FA Integration.....	1283
Outage System Integration.....	1288
The Big Picture of Outage System Integration.....	1288
Setting Up The System To Enable Outage Integration.....	1289
Business Flags.....	1294
Service Point Business Flags.....	1294
Customer Relationship Management Integration.....	1296
About Customer Relationship Requests.....	1296
Setting Up Customer Relationship Request Configuration.....	1296
Maintaining Customer Relationship Requests.....	1297
Control Table Setup Sequence.....	1297
Core Control Table Setup Sequence.....	1297
Cross-Reference To The Remaining Chapters.....	1313
Open-Item Accounting Table Setup Sequence.....	1314
Fund Accounting Table Setup Sequence.....	1314
Payment Event Distribution Table Setup Sequence.....	1314
Loans Table Setup Sequence.....	1314
Quotes Table Setup Sequence.....	1314
Non-billed Budget Table Setup Sequence.....	1314
Appointments Table Setup Sequence.....	1314
Scripting Table Setup Sequence.....	1315
Reports Setup Sequence.....	1315
XML Application Integration Setup Sequence.....	1315
Case Management Setup Sequence.....	1315
Workforce Management Setup Sequence.....	1315
Umbrella Agreement Management Setup Sequence.....	1315
Outage Management Setup Sequence.....	1315
Prepaid Metering Setup Sequence.....	1315
Batch Scheduler Setup Sequence.....	1316
Zone Set Up.....	1316
To Do Options Setup.....	1316
The Conversion Tool.....	1316
Introduction.....	1316
Conversion Tool Steps.....	1318
Map Legacy Data Into Staging Tables.....	1318
Validate Information In The Staging Tables.....	1319
Tidy Balances.....	1322
Balance Control (a).....	1324
Clear FT Balance Control.....	1324
Allocate Production Keys.....	1324
Insert Production Data.....	1327
Run Balance Control Against Production.....	1328
Validate Production.....	1328
The Validation User Interface.....	1328
Validation Error Summary.....	1328
Validation Error Detail.....	1329
FK Validation Summary.....	1329
FK Validation Detail.....	1329
The Staging Tables.....	1330
A Note About Programs in the Table Names Matrices.....	1330
Master Data.....	1331
Transaction Data.....	1348
Program Dependencies.....	1378

Appendix A - Entity Relationship Diagramming Standards.....1379
 Relationships..... 1380
 Entity..... 1380
 Color Coding..... 1380
Appendix B - Multiple Owners In A Single Database..... 1381
Appendix C - Known Oddities..... 1383

Chapter 1

Administrative User Guides

Oracle Utilities Customer Care and Billing is a customer information system designed to meet the needs of electric, gas, and water utilities, and comprises of the following key functional areas:

- **Customer Information** - Create and maintain demographic, geographic, and contract information. Start and stop customer service. Capture customer interaction.
- **Sales and Marketing** - Create and maintain initiatives, marketing campaigns, and packages that can be offered to customers. Add eligibility criteria, as necessary, in order to target specific customer segments. Create and maintain leads and orders that can market and enroll eligible customers into initiatives and packages.
- **Device Management** - Create and maintain meter and item details including installation and removal information. Generate field work to initiate meter and item tests, and record their subsequent results.
- **Field Work** - Create and maintain field work requests. Dispatch field work to appropriate parties and record their outcome.
- **Meter Reading** - Generate and download meter read routes to meter readers. Upload meter read details and track of consumption trends to highlight anomalies.
- **Billing and Statements** - Create and maintain bills. Recalculate historical bills, as needed. Consolidate financial information from multiple accounts into a single statement.
- **Rates** - Configure and maintain rates that will be used to calculate charges that appear on bills.
- **Payments** - Post and maintain customer payments. Distribute payments based on balance-forward or open-item accounting practices. Manage automatic payments, as appropriate.
- **Credit and Collections** - Monitor customer debt. Create and maintain processes to encourage customers to pay overdue debt. Create and maintain payment arrangements and pay plans. Write-off debt, if necessary.

This documentation describes the features and functionality of the Oracle Utilities Customer Care and Billing system.

Framework Administrative User Guide

The topics in this section describe how to administer the Oracle Utilities Application Framework.

Defining General Options

This section describes control tables that are used throughout your product.

Defining Installation Options

The topics in this section describe the various installation options that control various aspects of the system.

Installation Options - Main

Select **Admin > General > Installation Options - Framework** to define system wide installation options.

Description of Page

The **Environment ID** is a unique universal identifier of this instance of the system. When the system is installed, the environment id is populated with a six digit random number. While it is highly unlikely that multiple installs of the system at a given implementation would have the same environment ID, it is the obligation of the implementers to ensure that the environment ID is unique across all installed product environments.

System Owner will be **Customer Modification**.

The **Admin Menu Order** controls how the various control tables are grouped on Admin.

- If you choose **Functional**, each control table appears under a menu item that corresponds with its functional area. Note, the [menu](#) that is used when this option is chosen is the one identified with a menu type of **Admin**.
- If you choose **Alphabetical**, each control table appears under a menu item that corresponds with its first letter, using a Roman alphabet. For example, the Language control table will appear under the L menu item entry.

NOTE: The **Alphabetical** option only supports the Roman alphabet. For languages that do not use the Roman alphabet, the recommendation is to configure the system for the **Functional** setting.

CAUTION: In order to improve response times, installation options are cached the first time they are used after a web server is started. If you change the Admin Menu Order and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. Refer to [Caching Overview](#) for information on how to clear the system login cache (this is the cache in which installation options are stored).

The **Language** should be set to the primary language used by the installation. Note that if multiple languages are supported, each user may define their preferred language.

The **Currency Code** is the default currency code for transactions in the product.

If your product supports effective dated characteristics on any of its objects, define the date to be used as the **Characteristic Default Date** on objects without an implicit start date. The date you enter in this field will default when new characteristics are added to these objects (and the default date can be overridden by the user).

Active Owner displays the owner of newly added system data (system data is data like algorithm types, zones, To Do types, etc.). This will be **Customer Modification** unless you are working within a development region.

Country and **Time Zone** represent the default country and time zone that should be used throughout the application.

CAUTION: In most implementations, the time zone defined here matches the database time zone. However, if there is some reason that the database time zone does not match the installation time zone, an implementation may configure a setting in the properties file to automatically convert data from the database time zone to the time zone defined here

when displaying dates. Note that when this property setting is defined, changes to the installation time zone will require the server and the thread pool workers to be restarted in order for the changes to take effect.

Turn on **Seasonal Time Shift** if your company requires seasonal time shift information to be defined. Note that this is currently only applicable to Oracle Customer Care and Billing > Interval Billing functionality.

Installation Options - Messages

Select **Admin > General > Installation Options - Framework** and the **Messages** tab to review or enter messages that will appear throughout the application when a given event occurs.

The **Message** collection contains messages that are used in various parts of the system. For each message, define the **Installation Message Type** and **Installation Message Text**. The following table describes the **Message Types** provided by the framework product and how they are used in the system. Your specific product may have introduced additional message types.

Message Type	How The Message Is Used
Company Title for Reports	This message appears as a title line on the sample reports provided with the system. Generally it is your company name. It is only used if you have installed reporting functionality and are using the sample reports (or have designed your reports to use this message).

Installation Options - Algorithms

Select **Admin > General > Installation Options - Framework** and the **Algorithms** tab to review or enter the algorithms that should be evoked when a given event occurs.

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

CAUTION: These algorithms are typically significant processes. The absence of an algorithm might prevent the system from operating correctly.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Validate Email Attachment	Optional	Algorithms of this type are used to validate the attachments for size and total count while sending attachments using the Email service. Click here to see the algorithm types available for this system event.
Geocoding Service	Optional	Algorithms of this type use Oracle Locator to retrieve latitude and longitude coordinates using address information. Click here to see the algorithm types available for this system event.
Global Context	Optional	Algorithms of this type are called whenever the value of one of the global context fields is changed. Algorithms of this type are responsible for populating other global context

System Event	Optional / Required	Description
Guaranteed Delivery	Optional	<p>values based on the new value of the field that was changed.</p> <p>Refer to Global Context Overview for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Ldap Import	Optional	<p>Algorithms of this type may be called by processes that receive incoming messages that should 'guarantee delivery'. Refer to Guaranteed Delivery for more information. The business service F1-GuaranteedDelivery may be used to invoke this plug-in spot.</p> <p>Click here to see the algorithm types available for this system event.</p>
Ldap Import Preprocess	Optional	<p>Algorithms of this type are called for operations on users, groups, and group memberships after they have been processed.</p> <p>Click here to see the algorithm types available for this system event.</p>
Next To Do Assignment	Optional	<p>Algorithms of this type are called to preprocess data retrieved from LDAP.</p> <p>Click here to see the algorithm types available for this system event.</p>
Reporting Tool	Optional	<p>This type of algorithm is used to find the next To Do entry a user should work on. It is called from the Current To Do dashboard zone when the user ask for the next assignment.</p> <p>Click here to see the algorithm types available for this system event.</p>
SMS Receive Service	Optional	<p>If your installation has integrated with a third party reporting tool, you may wish to allow your users to submit reports on-line using report submission or to review report history online. This algorithm is used by the two on-line reporting pages to properly invoke the reporting tool from within the system.</p> <p>Click here to see the algorithm types available for this system event.</p>
SMS Send Service	Optional	<p>This type of algorithm is used to provide SMS receive service. Only one algorithm of this type should be plugged in.</p> <p>Click here to see the algorithm types available for this system event.</p>
To Do Information	Optional	<p>This type of algorithm is used to provide SMS send service. If your installation uses the base algorithm that uses BPEL, you will need to create a feature configuration with the SMS Send Configuration feature type to define your Oracle BPEL server and service call details. If your installation has integrated with a third-party SMS service, you may want to override this algorithm type with your own implementation. Only one algorithm of this type should be plugged in.</p> <p>Click here to see the algorithm types available for this system event.</p>
		<p>We use the term To Do information to describe the basic information that appears throughout the system to describe a To Do entry.</p>

System Event	Optional / Required	Description
		Plug an algorithm into this spot to override the system default "To Do information". Click here to see the algorithm types available for this system event.
To Do Pre-creation	Optional	These types of algorithms are called when a To Do entry is being added. Click here to see the algorithm types available for this system event.

Installation Options - Accessible Modules

Select **Admin > General > Installation Options - Framework** and the **Accessible Modules** tab to view the list of accessible modules.

Description of Page

This page displays the full list of the application's function modules. A **Turned Off** indication appears adjacent to a module that is not accessible based on your system's module configuration setup.

FASTPATH: Refer to [Module Configuration](#) for more information on function modules and how to turn off modules that are not applicable to your organization.

Installation Options - Installed Products

Select **Admin > General > Installation Options - Framework** and the **Installed Products** tab to view a read only summary of the products that are installed in the application version that you are logged into.

Description of Page

The **Product Name** indicates the name of the "products" that are installed. The collection should include **Framework**, an entry for your specific product and an entry for **Customer Release**.

Release ID shows the current release of the application that is installed. This field is used by the system to ensure that the software that executes on your application server is consistent with the release level of the database. If your implementation of the product has developed implementation-specific transactions, you can populate the Release Id for the **Customer Release** entry to define the latest release of implementation-specific logic that has been applied to this environment. In order for this to work, your implementation team should populate this field as part of their upgrade scripts.

The **Release ID Suffix**, **Build Number** and **Patch Number** further describe the details of your specific product release.

The **Display** column indicates the product whose name and release information should be displayed in the title bar. Only one product sets this value to **Yes**.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

Product Type indicates if the product is a Parallel Application. A parallel application is one that is independent of, and does not conflict with, other parallel applications. Multiple parallel applications can be installed in the same database and application server.

NOTE: About Information. The information on this tab is used to populate the information displayed in the [About](#) information for your product.

Support For Different Languages

User Language

The system provides support for multiple languages in a single environment. System users can use the system in their preferred language, as long as a translation into that language has been provided. A user sees the system in the language defined on their [user record](#). If enabled, users can use the [Switch Language](#) zone to switch to another supported language real time.

NOTE: Normally, setting up the system for another language is an implementation issue, not an administrative setup issue. However, there are several online administrative features that are used to set up a new language, and these are described here.

The following steps are required to support a new language:

- 1. Define a language code and indicate that it is enabled.** For details on this procedure, see [Defining Languages](#).
- 2. Copy descriptions of all language-enabled tables from an existing translation (e.g., English).** The copied values act merely as placeholders while the strings are translated into the new language. It is necessary to do this as a first step in order to create records using the new language code created in the previous step. Language-based descriptions can be copied using a supplied batch process, [FI-LANG](#). The batch copies all English labels in the system.
- 3. Apply the language pack.** If the product supplies a language pack with translations for the system metadata descriptions, follow the instructions provided with the language pack to add the translated text.
- 4. Translate additional content.** Translatable descriptions and labels for implementation data may be updated / entered in the application. First the user record must be updated to reference the new language. This may be done in one of the following ways:
 - a.** Switch to the new language using the [Switch Language](#) zone.
 - b.** If that zone is not available, navigate to the user page, assign the new language code to your User ID, sign out, and sign back in again.

Any online functions that you access will use your new language code. You can change the language code for all users who plan to use/modify the new language.

Customer Language

Your specific product may also support capturing the language of a customer. Such that correspondence sent from the product may be produced in a language set on a customer record. Refer to your specific product's documentation for more information about additional language support.

Defining Languages

Your product may support multiple languages. For example, the field labels, input text, and even outputs and reports can be configured to appear in a localized language. A language code for every potential language exists in the system to supply this information in various languages.

Select **Admin > General > Language** to define a language.

Description of Page

Enter a unique **Language Code**. If you are applying a language pack provided by the product, use the language code designed by the language pack.

Enter the **Description** for the language. Typically this should be the name of the language in that language.

Turn on **Language Enable** if the system should add a row for this language whenever a row is added in another language. For example, if you add a new currency code, the system will create language specific record for each language that has been enabled. You would only enable multiple languages if you have users who work in multiple languages. Languages that are configured as enabled, appear in the [Switch Language](#) dashboard zone. In addition, the login page for the application displays all the languages that are enabled, allowing the user to toggle the login instructions in that language.

NOTE: The list of enabled languages is captured on the server at startup time. If a new language is enabled, contact your server administrator to refresh the server in order to see the new language displayed in the login page.

The following two fields control how the contents of grids and search results are sorted by the Java virtual machine (JVM) on your web server:

- The **Locale** is a string containing three portions:
 - ISO language code (lower case, required)
 - ISO country code (upper case, optional)
 - Variant (optional).
- Underscores separate the various portions, and the variant can include further underscores to designate multiple variants. The specific JVM in use by your particular hardware/OS configuration constrains the available **Locales**. Validating the **Locale** against the JVM is outside the scope of this transaction. This means you are responsible for choosing valid **Locales**.

The following are examples of valid locales:

- en_US (this is for American English)
- en_AU (this is for Australian English)
- pt_BR (this is for Brazilian Portuguese)
- fr_FR_EURO (this is for European French)
- ja_JP (this if for Japanese)

In addition, the Java collation API can take a **Collator Strength** parameter. This parameter controls whether, for example, upper and lower-case characters are considered equivalent, or how accented characters are sorted. Valid values for collator strength are **PRIMARY**, **SECONDARY**, **TERTIARY**, and **IDENTICAL**. If you leave this field blank, Java will use its default value for the language. We'd like to stress that the impact of each value depends on the language.

Please see <http://java.sun.com/j2se/1.3/docs/guide/intl/locale.doc.html> for more information about the collator strength for your language.

Display Order indicates if this language is written **Left to Right** or **Right to Left**.

Owner indicates if this language is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a language. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_LANGUAGE](#).

Note that all administrative control tables and system metadata that contain language-specific columns (e.g., a description) reference a language code.

In addition, other tables may reference the language as a specific column. For example, on the User record you indicate the preferred language of the user.

Defining Countries

The topics in this section describe how to maintain countries.

Country - Main

To add or review Country definitions choose **Admin > General > Country**.

The **Main** page is used to customize the fields and field descriptions that will be displayed everywhere addresses are used in the system. This ensures that the all addresses conform to the customary address format and conventions of the particular country you have defined.

Description of Page

Enter a unique **Country** and **Description** for the country.

The address fields that appear in the **Main** page are localization options that are used to customize address formats so that they conform to address requirements around the world. By turning on an address field, you make that field available everywhere addresses for this country are used in the system. You can enter your own descriptions for the labels that suffix each switch; these labels will appear wherever addresses are maintained in the system.

NOTE: For any country where the **State** switch is checked, the valid states for the country must be entered on the **Country - State** tab. When entering address constituents on a record that captures this detail, the value for State is verified against the data in the State table. For any country where there is a component of the address that represents a "state" but your implementation does not want to populate the valid states for that country, choose a different field such as County for this constituent (and define an appropriate label). When entering address constituents on a record that captures this detail, no validation is done for the County column.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COUNTRY](#).

Country - States

To maintain the states located in a country, choose **Admin > Country > Search** and navigate to the **State** page.

Description of Page

For any country where you have enabled the State switch, use the **State** collection to define the valid states in the **Country**.

- Enter the standard postal abbreviation for the **State** or province.
- Enter a **Description** for this state or province.

Defining Currency Codes

The currency page allows you to define display options related to currency codes that are used by your system. Use **Admin > General > Currency** to define the currency codes in which financial information is denominated.

Description of Page

Enter a unique **Currency** and **Description** for the currency.

Use Currency **Symbol** to define the character that prefixes currency amounts in the system (e.g., \$ for U.S. dollars).

Enter the number of **Decimals** that will appear in the notation for the currency.

NOTE: Please contact your specific product to verify whether it supports a currency with more than 2 decimals.

The **Currency Position** indicates whether the currency symbol should be displayed as a **Prefix** or a **Suffix** to the currency amount.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CURRENCY_CD](#).

Defining Time Zones

The following topics describe how to design and set up time zones.

Designing Time Zones

NOTE: Oracle Utilities Customer Care and Billing - Interval Billing applications customers should consult the topic *Time Issues* (search the Help index for "time issues") for specific information relating to that product's interval billing time related functionality.

It is recommended that all time sensitive data is stored in the standard time (also called 'physical time') of the base time zone as defined on the installation options. This will prevent any confusion when analyzing data and will ensure that your algorithms do not have to perform any shifting of data that may be stored in different time zones.

The Time Zone entity is used to define all the time zones where your customers may operate. Each time zone should define an appropriate Time Zone Name. This is a reference to an external source that defines time zones, their relationship to Greenwich Mean Time, whether the time zone follows any shifting for summer / winter time (daylight savings time) and when this shift occurs.

When designing your time zones, the first thing to determine is the base time zone. You may choose the time zone where the company's main office resides. Once this is done you can link the time zone code to the installation option as the base time zone. Refer to [Installation Options - Main](#) for more information.

NOTE: An attribute in the system properties file may be configured to indicate that the DB session time zone should be synchronized with the value defined on the Installation Options. Refer to the *Server Administration Guide* for more information.

If your company does business beyond your main office's time zone, define the other time zones where you may have customers or other systems with which you exchange data. At this point, your specific product may include configuration tables to capture default time zones, for example based on a postal code or geographic location.

NOTE: Date and time in business object schemas. When defining date / time fields in a BO schema, schema attributes can be used to define whether or not data should be stored in standard time for the base time zone or if it should be stored in the standard time of another time zone (related to the data). In addition, schema attributes can be used to indicate if the display of the time should be shifted to represent the "local time". This is used to adjust for seasonal time differences. For example, if the data is stored in the appropriate time zone, but currently daylight savings time is being observed, the data will be shifted and shown in the "local" time. In addition, if the data is stored in the base time zone but the data is related to a different time zone, the data will be shown in the time zone appropriate for the data (including the appropriate seasonal adjustment). Refer to Schema Tips on the business object page for more information.

Setting Up Time Zones

Refer to [Designing Time Zones](#) for background information about defining time zones.

Open **Admin > General > Time Zone > Search** to define the time zones and their relation to the base time.

Description of Page

Enter a unique **Time Zone** and **Description** for the time zone.

Select the **Time Zone Name** from the list of Olson time zone values. This value is a reference to an external definition that allows the system to know how the time zone relates to Greenwich Mean Time and information about whether the time zone shifts for summer / winter time and when.

NOTE: The list includes abbreviated time zones that are not recommended and not supported by all functionality that accesses the time zone. The time zones in the 'area/location' format are the recommended values.

Indicate the **Shift in Minutes** that this time zone differs from the base time zone defined on the Installation Options. This is only applicable for the *Oracle Utility Customer Care and Billing - Interval Billing* application.

Indicate the **Seasonal Time Shift** applicable for this time zone. This is only applicable for the *Oracle Utility Customer Care and Billing - Interval Billing* application.

Default Time Zone Label and **Shifted Time Zone Label** are used for data that is sensitive to time zones and time shifting. It indicates whether the data displayed or data to be input is related to the "standard" time or the "shifted" time. For example, on a day when clocks are turned back one hour, a time entry of 1:30 a.m. needs to be labeled as either 1:30 a.m. standard time or 1:30 a.m. daylight savings time.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TIME_ZONE](#).

Setting Up Seasonal Time Shift

NOTE: The information in this topic applies only to Oracle Utilities Customer Care and Billing - Interval Billing applications.

Open **Admin > General > Seasonal Time Shift > Search** to define the seasonal time shift schedule.

Description of Page

Enter a unique **Seasonal Time Shift** code and **Description** for the seasonal time shift.

The Collection defines the **Effective Date/Time** (in standard time) that a time zone may shift in and out of standard time. If time is changed from standard time on the effective date/time, enter the **Shift in Minutes** that the time changes from standard time (usually **60**). If the time is changed back to standard time on the effective date/time, enter a **Shift in Minutes** of **0**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SEAS_TM_SHIFT](#).

Defining Geographic Types

If your company uses geographic coordinates for dispatching or geographic information system integration, you need to setup a geographic (coordinate) type for each type of geographic coordinate you capture on your premises and/or service points (geographic coordinates can be defined on both premises and service points).

To define geographic types, open **Admin > Geographic > Geographic Type**.

NOTE: Product specific. There is no framework functionality that uses this information. Refer to your specific product documentation to verify how this table is used in your specific product. In addition, use the data dictionary link below to determine if this object is a foreign key on any tables specific to your product.

Description of Page

Enter an easily recognizable **Geographic Type** code and **Description**.

Define the algorithm used to validate the **Validation Format Algorithm**. If an algorithm is specified, the system will validate that the geographic location entered on the premise and/or service point for the geographic type is in the format as defined in the algorithm. If you require validation, you must set up this [algorithm](#) in the system.

Click [here](#) to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_GEO_TYPE](#).

Defining Work Calendar

Workday calendars are used to ensure system-calculated dates fall on a workday. Select **Admin > General > Work Calendar > Search** to define a workday calendar.

Description of Page

The information on this transaction is used to define the days of the week on which your organization works.

Enter a unique **Work Calendar** and **Description**.

Turn on (check) the days of the week that are considered normal business days for your organization.

Use the collection to define the **Holiday Date**, **Holiday Start Date**, **Holiday End Date**, and **Holiday Name** for each company holiday. Holiday Start Date and Holiday End Date define the date and time that the holiday begins and ends. For example, your organization might begin a holiday at 5:00 p.m. on the day before the actual holiday.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CAL_WORK](#).

Defining Display Profiles

When you set up your [users](#), you reference a display profile. A user's display profile controls how dates, times, and numbers displayed. Choose **Admin > General > Display Profile > Search** to maintain display profiles.

Description of Page

Enter a unique **Display Profile ID** and **Description** to identify the profile.

Enter a **Date Format**. This affects how users view dates and how entered dates are parsed. This is a "free format" field with some rules.

- **dd** or **d** is interpreted as the day of the month. The **d** option suppresses a leading 0.
- **MM** or **M** is interpreted as the month number. The **M** option suppresses a leading 0.
- **yyyy**, **yy**, or **y** is interpreted as the year. The year can be 4 or 2 digits. The **y** option allows entry in either 2 or 4-digit form and is displayed in 2-digit form.
- Other characters are displayed as entered. Typically, these other characters should be separators, such as "-", ".", or "/". Separators are optional; a blank space cannot be use.

Here are some examples of date formats.

Format	Displayed / entered as
MM-dd-yyyy	04-09-2001
d/M/yyyy	9/4/2001
yy.MM.dd	01.04.09
MM-dd-y	04-09-01 - in this case you could also enter the date as 04-09-2001

NOTE: For centuries, the default pivot for 2-digit years is **80**. Entry of a 2-digit year greater than or equal to **80** results in the year being interpreted as 19xx. Entry of a 2-digit year less than **80** results in the year being interpreted as 20xx.

In addition, the following date localization functionality is supported. Note that in every case, the date is stored in the database using the Gregorian format. The settings below result in a conversion of the date for the user interface.

- **Hijri Dates**

Entering **iiii** for the year is interpreted as a year entered and displayed in Hijri format. For example, the Gregorian date 2014-05-30 may be entered / displayed as 1435/07/30 for a user whose display profile date format is **iiii/MM/dd**.

Note that this functionality relies on date mapping to be defined in the Hijri to Gregorian Date Mapping entry. Refer to [Additional Hijri Date Configuration](#) for more information.

- **Taiwanese Dates**

Entering **tttt** for the year is interpreted as a year entered and displayed in Taiwanese format where year 1911 is considered year 0000. For example, if the Gregorian date is 01-01-2005, it is displayed as 01-01-0094 for a user whose display profile date format is **dd-mm-tttt**.

- **Japanese Dates**

There are two options available for configuring Japanese Era date support. The setting **Gyy** for the year is interpreted as a year entered and displayed using an English character for the era followed by the era number. The letter 'T' is used for dates that fall within the *Taisho* era. The letter 'S' is used for dates that fall within the *Showa* era and the letter 'H' is used for dates that fall within the *Heisei* era. For example, for a user whose display profile date format is **Gyy/mm/dd** the Gregorian date 2008/01/01 is shown as **H20/01/01** ; the Gregorian date 1986/03/15 is shown as **S61/03/15**. The setting **GGGGyy** is interpreted as a year entered and displayed using Japanese characters for the era followed by the era number.

Japanese date limitations are as follows:

- The years 1912 through the current date are supported.
- Any functionality that displays Month and Year does not support Japanese Era dates. These dates are shown in Gregorian format.
- Graphs that display dates do not support the GGGGyy format.

Enter a **Time Format**. This is a "free format" display with some rules.

- **hh** or **h** is interpreted as the hour, 1-12; **KK** or **K** is interpreted as the hour, 0-11; **HH** or **H** is interpreted as the hour, 0-23; **kk** or **k** is interpreted as the hour, 1-24. The **h**, **K**, **H**, and **k** options suppress a leading 0.
- **mm** or **m** is interpreted as the minutes. The **m** option suppresses a leading 0.
- **ss** or **s** is interpreted as the seconds. The **s** option suppresses a leading 0.
- **a** is interpreted to mean display **am** or **pm** (only needed when the hour is entered in **hh**, **h**, **KK** or **K** formats). If an **am** or **pm** is not entered, it defaults to **am**.

Here are some examples of time formats.

Format	Displayed / entered as
hh:mma	09:34PM (can be entered as 09:34p)
hh:mm:ss	21:34:00
h:m:s	9:34:0

There are several options for displaying Numbers.

Decimal Symbol defines the separator between the integer and decimal parts of a number. Valid values are "." (a period) or "," (a comma),

Group Symbol defines the means to separate groups of bigger numbers. Valid values are

- ",", (comma). Large numbers group by threes separated by a comma, for example 1,000,000.

- "." (period). Large numbers group by threes separated by a period, for example 1.000.000.
- **None**. Large numbers do not have any separator, for example 1000000.
- **South Asian**. This option uses a comma for its separator but will group large numbers as follows: the first comma is used for the thousands separation and numbers over 9,999 are grouped with 2 units, for example 10,00,000.
- **Space**. Large numbers group by threes separated by a space, for example 1 000 000.

Negative Format defines how negative values are displayed. Valid values are **-9.9**, **(9.9)**, or **9.9-**.

Currency values can have a different **Negative Format** from other numbers. Valid values are **-S9.9**, **(S9.9)**, or **S9.9-**, where the "S" represents the currency symbol.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DISP_PROF](#).

Additional Hijri Date Configuration

For implementations that wish to support displaying dates according to the Hijri calendar, besides appropriate configuration in the [Display Profile](#), the mapping between the Hijri dates and the Gregorian dates must be entered. This mapping is defined in the [Hijri to Gregorian Date Mappingmaster configuration](#) record.

The mapping record contains a collection of entries for each year in the Islamic calendar.

For each year, clicking the Expand Zone icon shows the mapping collection with the first date of each month of the Hijri calendar. The corresponding date in the Gregorian calendar should be entered for each row.

Defining Phone Types

Phone types define the format for entering and displaying phone numbers.

To add or review phone types, choose **Admin > General > Phone Type**.

Description of Page

Enter a unique **Phone Type** and **Description** for each type of phone number you support.

Select an appropriate **Phone Number Format Algorithm** for each **Phone Type**. This algorithm controls the format for entry and display of phone numbers. Click [here](#) to see the algorithm types available for this plug-in spot.

Use **Phone Type Flag** to define if this type of phone number is a **Fax** number. Defining which phone type is used for facsimile transmittal is only pertinent if your product supports routing of information via fax. For example, in Oracle Utilities Customer Care and Billing, the system may be configured to fax a bill to a customer.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PHONE_TYPE](#).

Setting Up Characteristic Types & Values

Many objects in the system support a collection of Characteristics, which are used to capture additional fields for the object that are not already supported by the object's provided attributes. Each characteristic is associated with a characteristic type, which defines attributes of the field you wish to capture.

All characteristics are captured as a list. However, the user interface for characteristics differ based on the type of page that is used to maintain the object.

- For portal based pages the business object drives the display and maintenance of an object. For these types of pages, it is recommended that characteristics are defined as part of the business object schema allowing the user interface to display

the characteristic as if it is another field. However, the display / maintenance of the characteristic is determined by the business object's user interface design.

- There are some fixed pages in the system that do not support customization of the user interface. For these objects, the characteristics are displayed / maintained as a generic list.

The topics in this section describe how to setup a characteristic type.

There Are Four Types Of Characteristics

Every characteristic referenced on an object references a characteristic type. The characteristic type controls the validity of the information entered by a user when they enter the characteristic's values. For example, if you have a characteristic type on user called "skills", the information you setup on this characteristic type controls the valid values that may be specified by a user when defining another user's skills.

When you setup a characteristic type, you must classify it as one of the following categories:

- **Predefined value.** When you setup a characteristic of this type, you define the individual valid values that may be entered by a user. A good example of such a characteristic type would be one on User to define one or more predefined skills for that user. The valid values for this characteristic type would be defined in a discreet list.
- **Ad hoc value.** Characteristics of this type do not have their valid values defined in a discreet list because the possible values are infinite. Good examples of such a characteristic type would be ones used to define a user's birth date or their mother's maiden name. Optionally, you can plug-in an algorithm on such a characteristic type to validate the value entered by the user. For example, you can plug-in an algorithm on a characteristic type to ensure the value entered is a date.
- **Foreign key value.** Characteristics of this type have their valid values defined in another table. For example perhaps you want to link a user to a table where User is not already a foreign key. Valid values for this type of characteristic would be defined on the user table. Please be aware of the following in respect of characteristics of this type:
 - Before you can create a characteristic of this type, information about the table that contains the valid values must be defined on the [foreign key reference table](#).
 - The referenced table does not have to be a table within the system.
 - Not all entities that support characteristics support foreign key characteristics. Refer to the data dictionary to identify the entities that include the foreign key characteristic columns.
 - As described in [Search Options](#), there are two different searching metaphors supported on FK reference. If the object that a characteristic is being linked to is defined on a fixed page, it will display a search icon if the characteristic type's FK reference defines a navigation key based search. If the object is maintained on a portal based page, it will display a search icon if the characteristic type's FK reference defines a search zone.
- **File Location.** Characteristics of this type contain a URL. The URL can point to a file or any web site. Characteristics of this type might be useful to hold references to documentation / images associated with a given entity. For example, the image of a letter sent to you by one of your customers could be referenced as a file location characteristic on a customer contact entry. When such a characteristic is defined on an entity, a button can be used to open the URL in a separate browser window. Note that for references to a file, the recommendation is to use the Attachment functionality to link documents to an object rather than a characteristic type of File Location. Refer to [Attachment Overview](#) for more information. The documentation related to file location remains for upgrade purposes.

Searching By Characteristic Values

For certain entities in the system that have characteristics, you may search for a record linked to a given characteristic value. The search may be done in one of the following ways:

- Some base searches provide an option to search for an object by entering Characteristic Type and Characteristic Value.

- Your implementation may define a customized search for an entity by a characteristic value for a specific characteristic type using a query data explorer.
- Your implementation may require a business service to find a record via a given characteristic value. For example, maybe an upload of user information attempts to find the user via an Employee ID, defined as a characteristic.

Not all entities that support characteristics support searching by characteristics. Refer to the data dictionary to identify the characteristic collections that include the search characteristic column.

CAUTION: For ad-hoc characteristics, only the first 50 bytes are searchable. For foreign key characteristics, the search value is populated by concatenating the values of each foreign key column to a maximum of 50 bytes.

For the base searches that provide a generic option to search by characteristic type and value, you can restrict the characteristic types that can be used to search for an entity. For example, imagine you use a characteristic to define a "jurisdiction" associated with a To Do for reporting purposes. If your company operates within a very small number of jurisdictions, you wouldn't want to allow searching for a To Do by jurisdiction, as a large number of To Do entries would be returned.

A flag on the [characteristic type](#) allows an administrator to indicate if searching by this characteristic type is **allowed** or **not allowed**.

Characteristic Type - Main

To define a characteristic type, open **Admin > General > Characteristic Type**.

Description of Page

Enter an easily recognizable **Characteristic Type** and **Description** for the characteristic type. **Owner** indicates if this characteristic type is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new characteristic type, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Use **Type of Char Value** to classify this characteristic type using one of the following options (refer to [There Are Four Types Of Characteristics](#) for more information):

- **Predefined value.** Characteristics of this type have their valid values defined in the **Characteristic Value** scroll, below. For each valid value, enter an easily recognizable **Characteristic Value** and **Description**.
- **Ad hoc value.** Characteristics of this type capture free form text. If you use this option, you can optionally define the **Validation Rule** used to validate the user-entered characteristic value. Click [here](#) to see the algorithm types available for this plug-in spot.
- **File location value.** Characteristics of this type contain a URL. The URL can point to a file or any web site. Note that for references to a file, the recommendation is to use the Attachment functionality to link documents to an object rather than a characteristic type of File Location. Refer to [Attachment Overview](#) for more information. The documentation related to file location remains for upgrade purposes.

File location characteristic values must be entered in a "non-relative" format. For example, if you want to define a characteristic value of *www.msn.com*, enter the characteristic value as `http://www.msn.com`. If you omit the `http://` prefix, the system will suffix the characteristic value to the current URL in your browser and attempt to navigate to this location when the launch button is pressed. This may or may not be the desired result.

NOTE:

Due to browser security restrictions, opening URLs using the file protocol ("file://") from pages retrieved using http does not work for Internet Explorer version 7 or later, or in Firefox. If the file protocol is used, the browser either does not return properly or an error is thrown (e.g., "Access Denied", which usually results from cross site scripting features added for security reasons).

This issue has no known workaround. To comply with browser security standards, the recommendation is to move the target files to an FTP or HTTP server location to avoid protocols that are subject to browser security restrictions.

- **Foreign key reference.** Characteristics of this type have their valid values defined in another table. If you choose this option, you must use **FK Reference** to define the table that controls the valid values of this characteristic type. Refer to [Setting Up Foreign Key References](#) for more information.

Use the **Allow Search by Char Val** to indicate if searching for an entity by this characteristic type is **Allowed** or **Not Allowed**. Refer to [Searching by Characteristic Values](#) for more information.

The **Custom** switch is only applicable to **Predefined value** types. It indicates whether or not an implementation is allowed to add values for a characteristic type whose owner is not **Customer Modification**

- If this switch is turned on, an implementation may add characteristic values to the grid for system owned characteristic types.
- If this switch is turned off, an implementation may not add characteristic values to the grid for system owned characteristic types.

NOTE: Regardless of the value of the Custom switch, an implementation may not update or remove system owned characteristic values.

The **Characteristic Value** grid defines the valid values for a **Predefined value** type of characteristic.

The **Characteristic Value** is the unique identifier of the value.

Description is the text that is visible in the dropdowns and display when viewing this characteristic value.

Owner indicates if this characteristic value is owned by the system or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add characteristic values to a characteristic type. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CHAR_TYPE](#) in the data dictionary schema viewer.

Characteristic Type - Characteristic Entities

To define the entities (objects) on which a given characteristic type can be defined, open **Admin > General > Characteristic Type** and navigate to the **Characteristic Entities** tab.

Description of Page

Use the **Characteristic Entity** collection to define the entities on which the characteristic type can be used. **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

NOTE: The values for this field are customizable using the Lookup table. This field name is **CHAR_ENTITY_FLG**.

NOTE: For some entities in the system, the valid characteristics for a record are defined on a related "type" entity. For example, the To Do type defines valid characteristics for manually created To Do entries of that type. When configuring your system, in addition to defining the appropriate entity for a characteristic type, you may also need to link the characteristic type to an appropriate entity "type". This technique is typically not followed for business object driven maintenance objects, where the business objects can be configured with the appropriate "flattened" characteristic types in the schema.

Setting Up Foreign Key Reference Information

A Foreign Key Reference defines the necessary information needed to reference an entity in certain table.

You need to set up this control table if you need to validate a foreign key value against a corresponding table. For example, if a schema element is associated with an FK Reference the system validates the element's value against the corresponding table. Refer to [Configuration Tools](#) to learn more about schema-based objects. Another example is characteristics whose valid values are defined in another table (i.e., you use "foreign key reference" characteristic types). Refer to [There Are Four Types Of Characteristics](#) for a description of characteristics of this type.

A FK Reference is used not just for validation purposes. It also used to display the standard information description of the reference entity as well as provide navigation information to its maintenance transaction. Info descriptions appear throughout the UI, for example, whenever an account is displayed on a page, a description of the account appears. The product provides base product FK references for many of its entities as they are used for validation and display of elements in both fixed page user interfaces as well as portal based user interfaces.

An implementation may also see the need to define a foreign key reference. The following points describe what you should know before you can setup a foreign key reference for a table.

- The physical name of the table. Typically this is the primary table of a maintenance object.
- The program used by default to construct the referenced entity's info description. Refer to [Information Description Is Dynamically Derived](#) for more information on how this is used.
- The transaction used to maintain the referenced entity. This is where the user navigates to when using the "go to" button or hyperlink associated with the entity. Refer to [Navigation Information Is Dynamically Derived](#) for more information on how this is used.
- The name of the search page used to look for a valid entity. Refer to [Search Options](#) for more information.

Information Description Is Dynamically Derived

Typically a FK Reference is defined for a maintenance object's primary table. In this case the system dynamically derives the standard information associated with a specific referenced entity as follows:

- Attempt to determine the business object associated with the referenced entity. Refer to the [Determine BO](#) maintenance object algorithm system event for more information. If a business object has been determined, the system lets the business object's [Information](#) plug-in, if any, format the description.
- If a business object has not been determined or the business object has no such plug-in, the system lets the maintenance object's [information](#) plug-in, if any, format the description.
- If the maintenance object has no such plug-in, the system uses the info program specified on the FK Reference to format the description.

NOTE: Technical note. The class that returns the information displayed adjacent to the referenced entity is generated specifically for use as an info routine. Please speak to your support group if you need to generate such a class.

NOTE: Generic routine. The system provides a generic information routine that returns the description of control table objects from its associated language table. By "control table" we mean a table with an associated language table that contains a **DESCR** field. Refer to [Defining Table Options](#) for more information on tables and fields. The java class is `com.splwg.base.domain.common.foreignKeyReference.DescriptionRetriever`.

Navigation Information Is Dynamically Derived

Typically a FK Reference is defined for a maintenance object's primary table. In this case the system dynamically derives the actual transaction to navigate to for a given referenced entity as follows:

- Attempt to determine the business object associated with the referenced entity. Refer to the [Determine BO](#) maintenance object algorithm system event for more information. If a business object has been determined, use the maintenance portal defined as its **Portal Navigation Option** business object option.
- If a business object has not been determined or the business object defines no such option, the system uses the transaction specified on the FK Reference.

Search Options

The product provides two main metaphors for implementing a user interface. For input fields that are foreign keys, search options are dependent on the metaphor used by the page in question.

- A [fixed maintenance page](#) user interface is a page supplied by the base product where only minor enhancements, if any, can be introduced by implementations. The foreign key reference may be used in one of two ways.
 - The based product may use an FK reference to define a base element on one of these pages. If a search is available for such elements, the FK reference's Search Navigation Key is used to implement the search.
 - Entities that support characteristics typically include a generic characteristic collection UI metaphor on these types of pages. In this metaphor, a foreign key characteristic displays a search icon if the FK Reference has configured a Search Navigation Key.
- A [portal based](#) user interface is a more flexible user interface where an implementation has more options for customizing the look and feel. The base product uses UI maps or automatic UI rendering to display input fields. Elements that are foreign keys may display a search icon if the FK reference defines a Search Zone.

NOTE: Defining search zones directly. It's possible for elements on a UI map to define a specific search zone directly in the HTML, rather than using the search zone defined on an FK reference. Refer to the UI map tips for more information on implementing searches using zones.

NOTE: Not every FK reference provided with the product is configured with both search options. Specifically, objects that are maintained in a portal based page typically do not provide a navigation key based search. It means that if linking this type of object as a characteristic to an object that is maintained on a fixed page, a search will not be available.

Foreign Key Reference - Main

To setup a foreign key reference, open **Admin > Database > FK Reference**.

CAUTION: Important! If you introduce a new foreign key reference, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Description of Page

Enter an easily recognizable **FK** (foreign key) **Reference** code and **Description** for the record.

Enter the name of the **Table** whose primary key is referenced. After selecting a **Table**, the columns in the table's primary key are displayed adjacent to **Table PK Sequence**.

Use **Navigation Option** to define the page to which the user will be transferred when they press the go to button or hyperlink associated with the referenced entity. Refer to [Navigation Information Is Dynamically Derived](#) for more information on how this is used.

The **Info Program Type** indicates whether the default program that returns the standard information description is **Java** or **Java (Converted)**, meaning it was converted into Java.

NOTE: **Java (Converted)** program types are not applicable to all products.

Use **Info Program Name** to enter the Java class / program name.

Refer to [Information Description Is Dynamically Derived](#) for more information on the info program is used.

NOTE: View the source. If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#).

Use **Context Menu Name** to specify the context menu that appears to the left of the value.

NOTE: Context Menu Name is not applicable to user interface elements displaying a generic collection using a foreign key characteristic type. It is only applicable for pages utilizing the foreign key compound element type for fixed page user interface and for data displayed in a portal based user interface where the foreign key reference is defined as an attribute for an element. Report parameters that reference foreign key characteristics are an example of a user interface where a context menu is not displayed even if the foreign key reference defines one.

Use **Search Zone** to define the search zone that opens when a user searches for valid values when the foreign key reference is configured as an input field on a portal based page. Refer to [Search Options](#) for more information.

Use **Search Navigation Key** to define the search page that will be opened when a user searches for valid values on a user interface that is a fixed page. Refer to [Search Options](#) for more information.

Use **Search Type** to define the default set of search criteria used by the **Search Navigation Key's** search page.

Use **Search Tooltip** to define a label that describes the **Search Navigation Key's** search page.

NOTE: Search Type and Search Tooltip. These attributes are only applicable to user interface elements utilizing the foreign key compound element type on fixed page user interfaces. Report parameters that reference foreign key characteristics are an example of a user interface where this information is not used even if the foreign key reference defines them.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_FK_REF](#).

Defining Feature Configurations

Some system features are configured by populating options on a "feature configuration". Because various options throughout the system may be controlled by settings in feature configuration, this section does not document all the disparate possible options. The topics below simply describe how to use this transaction in a generic way.

For information about specific features:

- Refer to the detailed description of each option type.
- Use the index in the online help and search for 'feature configuration' to find any specific topics describing feature options in the administration guide.

You can create options to control features that you develop for your implementation. To do this:

- Review the lookup values for the lookup field **EXT_SYS_TYP_FLG**. If your new option can be logically categorized within an existing feature type, note the lookup value. If your new option warrants a new feature type, add a lookup value to this lookup field.
- Define the feature's option types. If you have identified an existing feature type to add the options to, find the lookup with the name **xxxx_OPT_TYP_FLG** where **xxxx** is the lookup value of **EXT_SYS_TYP_FLG** noted above. If you decided to create a new feature type (by adding a new lookup value to the **EXT_SYS_TYP_FLG** lookup, you must create a new lookup with the name **xxxx_OPT_TYP_FLG** where **xxxx** is the new value you defined above.
- Flush all caches.

Feature Configuration - Main

To define your feature configuration, open **Admin > General > Feature Configuration**.

Description of Page

Enter an easily recognizable **Feature Name** code.

Indicate the **Feature Type** for this configuration. For example, if you were setting up the options for the external messages, you'd select **External Messages**.

NOTE: You can add new Feature Types. Refer to the description of the page above for how you can add Feature Types to control features developed for your implementation.

NOTE: Multiple Feature Configurations for a Feature Type. Some Feature Types allow multiple feature configurations. The administration documentation for each feature will tell you when this is possible.

The **Options** grid allows you to configure the feature. To do this, select the **Option Type** and define its **Value**. Set the **Sequence** to **1** unless the option may have more than value. **Detailed Description** may display additional information on the option type.

NOTE: Each option is documented elsewhere. The administration documentation for each feature describes its options and whether an option supports multiple values. Use the index to look for 'feature configuration' to find the various types of feature options.

NOTE: You can add new options to base-package features. Your implementation may want to add additional options to one of the base-package's feature types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do this, display the lookup field that holds the desired feature's options. The lookup field's name is **xxxx_OPT_TYP_FLG** where **xxxx** is the identifier of the feature on the **EXT_SYS_TYP_FLG** lookup value. For example, to add new batch scheduler options, display the lookup field **BS_OPT_TYP_FLG**.

Feature Configuration - Messages

If the feature exists to interface with an external system, you can use this page to define the mapping between error and warning codes in the external system and our system.

Open this page using **Admin > General > Feature Configuration** and navigate to the **Messages** tab.

Description of Page

For each message that may be received from an external system, define the **Feature Message Category** and **Feature Message Code** to identify the message.

A corresponding message must be defined in the [system message](#) tables. For each message identify the **Message Category** and **Message Number**. For each new message, the Message Category defaults to **90000** (because an implementation's messages should be added into this category or greater so as to avoid collisions during upgrades).

Defining Master Configurations

A master configuration is an object that enables an implementation to define configuration for features in the system. It is an alternative to using feature configuration for defining options. A master configuration is defined using a business object. Only one master configuration may exist for a given business object.

The product provides one or more master configuration that may be used for configuration. Some examples, of base master configuration business objects are as follows

- **Hijri to Gregorian Date Mapping.** This allows an implementation that uses Hijri dates to define the mapping between those dates and Gregorian dates.
- **ILM Configuration.** For implementations that use Information Lifecycle Management, the ILM configuration record defines some parameters used by the process.
- **Migration Assistant Configuration.** For implementations that use the configuration migration assistant (CMA), the configuration record defines some parameters used by the process.

For a list of all the master configuration records provided by the product, navigate to the master configuration page in the application. To find help topics related to functionality controlled by the master configuration records, use the keyword 'master configuration' in the index.

To set up a master configuration, open **Admin > General > Master Configuration**.

The topics in this section describe the base-package zones that appear on the Master Configuration portal.

Master Configuration

The Master Configuration List zone lists every category of master configuration.

The following functions are available:

- If a master configuration record exists for a given master configuration business object, the broadcast icon may be used to view details information about the adjacent master configuration. In addition, an edit icon is visible to allow a user to update the record.
- If a master configuration record does not exist for a given master configuration business object, the add icon is visible to allow a user to define the record.

Master Configuration Details

The Master Configuration Details zone contains display-only information about a master configuration.

This zone appears when a master configuration has been broadcast from the Master Configuration zone.

Please see the zone's help text for information about this zone's fields.

Defining Security & User Options

The contents of this section describe how to maintain a user's access rights.

The Big Picture of Application Security

The contents of this section provide background information about application security.

Application Security

The system restricts access to its transactions as follows:

- An [application service](#) may be associated with every securable function in the system.
 - All maintenance objects define an application service that includes the basic actions available, typically **Add, Change, Delete, and Inquire**. The base product supplies an application service for every maintenance object.
 - For maintenance objects whose user interface page is not portal-based, the application service also controls whether the menu entry appears. If a user doesn't have access to the maintenance object's application service, the menu item that corresponds with the application service will not be visible.
 - For portal based user interfaces, each main portal defines an explicit application service with the access mode **Inquire**, allowing the user interface to be secured independently of the underlying object security. If a user doesn't have access to the portal's application service, the menu item that corresponds with the application service will not be visible. The base product supplies an application service for every portal that is accessible from the menu.
 - Menu items may define an application service. Use this technique for the following scenarios:
 - Suppress a menu item if the underlying application security for the transaction does not provide enough fine grained control. For example, imagine your implementation creates a special BPA script to add a To Do Entry and would like users to use the special BPA rather than the base supplied Add dialogue for To Do Entry. The underlying security settings for To Do Entry should grant Add access to these users given that the special BPA will still add a record. To suppress the base Add dialogue, link a special application service and access mode for the base supplied menu item for To Do Entry Add. Then define a menu entry for the new special BPA for adding.
 - Suppress the add option if a user does not have add security for the object. By default the product does not suppress the add function if a user does not have add access to the object. Rather, the user is prevented from adding the record at the back-end. If your implementation would like to suppress the icon, link the object's application service and the Add access mode to the Add menu item.
 - Zones define an application service
 - For zones linked to a portal, if a user doesn't have access to the zone's application service, the zone will not be visible on the portal. In most cases the zone defines the same application service as its portal. In special cases, such as the zones on the Dashboard, the product supplies separate application services for each zone allowing implementations to determine at a more granular level which users should have access to which zones.
 - For query zones that are configured on a multi-query zone, if a user doesn't have access to the zone's application service, the zone will not be visible in the dropdown on the multi-query zone. In most cases all zones in a multi-query zone define the same application service as the multi-query zone. The product may supply a special application service for one or more zones in a multi-query zone if the functionality is special to certain markets or jurisdictions and not applicable to all implementations.
 - For zones that are used by business services to perform SQL queries, the product supplies a default application service. Security for these zones is not checked by the product as they are used for internal purposes.
 - Business objects define an application service. If the business object defines a lifecycle, the application service must include access modes that correspond to each state. In addition, the standard maintenance object access modes of **Add, Change, Delete and Inquire** are included. The base product business objects are supplied with appropriate application services.
 - Other configuration tool objects are securable but the base product typically does not supply special application services for each object. An implementation may supply custom application services and link them to the appropriate record:
 - BPA scripts may define an application service with the access mode **Execute**. The base BPA scripts are typically not configured with any application service. An implementation may define one. Note that as mentioned above, a menu item may also be configured with an application service and access mode. This allows for a BPA that is invoked via a menu entry to be secured in more than one way.
 - Business Services and Service Scripts define an application service with the access mode **Execute**. This is needed for services that may be executed from an external system, for example via an inbound web service. Base business services and service scripts that are linked to an inbound web service are configured with special application service. All other business services and service scripts are delivered with a default application service, which may be overridden by an implementation.

- Users are granted access to application services via [user groups](#). For example, you may create a user group called Senior Management and give it access to senior manager-oriented pages and portals.
- When you grant a user group access to an application service with multiple access modes, you must also define the access modes that are allowed. Often the access modes correspond to an action on a user interface. For example, you may indicate a given user group has **inquire**-only access to an application service, whereas another user group has **add, change, cancel** and **complete** access to the same service. Refer to [action level security](#) for more information.
- If the application service has [field level security](#) enabled, you must also define the user group's security level for each secured field on the transaction.
- And finally, you link individual [users](#) to the user groups to which they belong. When you link a user to a user group, this user inherits all of the user group's access rights.

CAUTION: Menus may be suppressed! If all menu items on a menu are suppressed, the menu is suppressed.

Action Level Security

When you grant a user group access to an [application service](#), you must indicate the actions to which they have access.

- For application services that only query the database, there is a single action to which you must provide access - this is called **Inquire**.
- For application services that can modify the database, you must define the actions that the user may perform. At a minimum, most maintenance transactions support **Add, Change, and Inquire** actions. Additional actions are available depending on the application service's functions.

CAUTION: Important! If an application service supports actions that modify the database other than **Add, Change, and Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a transaction that supports special actions in addition to **Add, Change, and Inquire** (e.g., **Freeze, Complete, Cancel**). If you want to give a user access to any of these special actions, you must also give the user access to the **Inquire and Change** actions.

Field Level Security

Sometimes transaction and action security is not sufficient. There are situations where you may need to restrict access based on the values of data. For example, in Oracle Utilities Customer Care and Billing you might want to prevent certain users from completing a bill for more than \$10,000. This is referred to as "field level security".

Field level security can be complex and idiosyncratic. Implementing field level security always requires some programming by your implementation group. This programming involves the introduction of the specific field-level logic into the respective application service(s).

NOTE: **Field level security logic is added to user exits.** Refer to the Public API chapter of the Software Development Kit Developer Guide for more information on how to introduce field-level security logic into an application service's user exits.

Even though the validation of a user's field-level security rights requires programming, the definition of a user's access rights is performed using the same transactions used to define transaction / action level security. This is achieved as follows:

- Create a [security type](#) for each type of field-level security.
- Define the various access levels for each security type. For example, assume you have some users who can complete bills for less than \$300, and other users who can complete bills for less than \$1,000, and still other users who can complete bills for any value. In this scenario, you'd need 3 access levels on this security type:
 - Level 1 (lowest): May authorize bills <= \$300
 - Level 2 (medium): May authorize bills <= \$1,000

- Level 3 (highest): May authorize all bills
- Link this security type to each [application service](#) where this type of field level security is implemented. This linkage is performed on the [security type](#) transaction.
- Defining each [user group's](#) access level for each security type (this is done for each application service on which the security type is applicable).

NOTE:

Highest value grants highest security. The system expects the highest authorization level value to represent highest security level. Moreover, authorization level is an alphanumeric field so care should be taken to ensure that it's set up correctly.

Encryption and Masking

"Encryption" refers to encrypting data stored in the database using an encryption key.

"Masking" refers to overwriting all or part of an un-encrypted field value with a masking character. For example, perhaps only the last 4 digits of a credit card number are visible with the other digits changed to an asterisk. The system provides support for masking in two ways:

- A field value is stored as plain text and is masked for the presentation layer only.
- If a field is encrypted, the encrypted data is stored in a special field. The field visible to the user interface is stored with a masked value.

The system provides the ability to define configuration to indicate that data should be encrypted or masked. The following sections provide more information about each feature.

User Interface Masking

The functionality described in this section is used to take data that is stored in plain text in the database and mask the value before it is presented to a user (or an external system). This feature includes the ability to allow some users to view the data unmasked using security configuration. The system allows different masking rules to be applied to different fields. For example, a credit card number can be masked differently than a social security number.

The following topics describe how to mask field values.

Identify the Data to be Masked

Identify the data that is stored as plain text, but should be masked for display to users. For example, imagine that you have identified that Credit Card Numbers and a person's federal ID number (for example, in the United States, the Social Security Number or SSN). Each field identified may be displayed and maintained in different user interfaces throughout the system, but the masking rules for a given field are probably uniform regardless of where the data is displayed.

Primary keys cannot be masked. A field defined as a unique identifier of a row cannot be configured for masking. Masking a field that is part of the primary key causes a problem when attempting to update the record. This restriction also applies to elements that are part of a "list" in an XML column on a maintenance object. One or more elements in the list must be defined as a primary identifier of the list. Be sure that primary key elements in the list are not ones that require masking.

List members that contain different "types". Consider a page with a list that contains a person's identification numbers. You can set up the system so that a person's social security number has different masking rules than their drivers license number. If your implementation has this type of requirement, the list of masked fields should contain an entry for each masking rule.

For each field, if there are some users that may see the data unmasked on one or more of the user interfaces, then security configuration is required. If the value of a field should be masked for all users across all pages in the application, then the security configuration is not needed.

Security Configuration

Define a [security type](#) for each field with two authorization levels:

- **1** - Can only see the element masked
- **2** - Can only see the element unmasked

Link all of the security types to an [application service](#) of your choosing. We recommend linking every masking-oriented security type to a single application service (e.g., **CM_MASK**) as it makes granting access easier.

For each security type, identify which users can see its data unmasked and which users can only see its data masked. If the masked and unmasked users fit into existing user groups, no additional user groups are necessary. Otherwise, create new user groups for the masked and unmasked users.

After the user groups for each security type are defined, [link each user group to the application service](#) defined above. When a user group is linked to the application service, you will define the authorization level for each security type linked to the application service. If a user group's users should see the security type's field values unmasked, set the authorization level to 2; otherwise set it to 1.

NOTE: Flush the cache. Remember that any time you change access rights you should [flush the security cache](#) (by entering flushAll.jsp on the URL of the application) if you want the change to take effect immediately.

Configure a Masking Algorithm

A data masking algorithm must be created for each combination of masking rules and security type. These algorithms determine if a user has the rights to view a given field unmasked, and, if not, how the field should be masked.

The base package provides the algorithm type **FI-MASK** whose parameters are designed to handle most masking needs. If certain users may see the data unmasked, parameters capture the application service, security type and authorization level defined above used to evaluate this. In addition, parameters allow you to configure how much of the data to mask, what masking character to use. Refer to the algorithm type description for more information.

Click [here](#) for a list of all the algorithms supplied for this plug-in spot.

Determine How the Fields are Displayed

The masking configuration differs based on how a field is retrieved for access to the user interface. So for the masking of one “logical” field (like a person’s SSN), there may be multiple configuration entries required to cover all the access methods. Review each user interface where a given field is displayed and create the following categories:

- The field is an element that is retrieved by invoking a business object, a business service, or a service script
- The field is displayed on a fixed maintenance page (and is therefore retrieved by invoking a page service)
- The field is displayed on a fixed search page (and is therefore retrieved by invoking a search service)
- The field is stored as an ad hoc characteristic

Create a Feature Configuration for Each Masked Element

Create a feature configuration with a Feature Type of **Data Masking**. An option entry with option type of **Field Masking** is needed for every combination of field to mask and the method used to display the data. The value will contain mnemonics that reference the appropriate data masking algorithm along with configuration that differs depending on how the field is retrieved for display as described below.

Schema Based Object Field Masking

For data that is accessed via a schema-based object call and displayed in a UI map, the field to be masked must reference a meta-data field name in its schema definition: **field="fld_name", alg="algorithm name"**

If the element references an mdField in the schema, that is the field used to identify the masking rule. If there is no mdField reference but only a mapField reference, that is the field used to identify the masking rule. For example, if you want to mask a credit card number, let's assume that field is defined in the schema is the following:

```
<creditCard mdField="CCNBR" mapField="EXT_ACCT_ID"/>
```

In this case, the option value should be **field="CCNBR", alg="algorithm name"**. An option value of **field="EXT_ACCT_ID", alg="algorithm name"** would not result in masking.

A "where" clause may also be specified. This is useful for data that resides in a list where only data of a certain type needs to be masked: **field="fld_name", alg="algorithm name", where="fld_name='value'"**

For example, person can have a collection of IDs and only IDs of type 'SSN' (social security number) should be masked. If the person data including its collection of person IDs is displayed on a UI map via a business object call, let's assume the collection is defined in the following way:

```
<personIds type="list" mapChild=CI_PER_ID">  
  <isPrimaryId mapField="PRIM_SW"/>  
  <idType mapField="ID_TYPE_CD"/>  
  <personIdNumber mapField="PER_ID_NBR"/>  
</personIds>
```

The option value may look like this: **field="PER_ID_NBR", alg="algorithm name", where="ID_TYPE_CD='SSN'"**

Please note the following important points for schema based masking:

- **Limitation of 'where' field** Although the main use of a 'where' clause for schema oriented elements is to mask certain elements in a list based on a 'type', it is also possible to mask a single field in the schema based on the value of another field. For example, imagine that a customer submits a registration form that defines an ID type and ID value. Although this data is not in a list, the implementation may still want to only mask the ID value if the ID type is "SSN". The framework is only able to mask an element in the schema based on a 'where' clause if the element in the 'where' clause is a "sibling" in the schema.
 - If the element to be masked is in a list, the element in the 'where' clause must be in the same list.
 - If an element to be masked maps to a real column in a table, the element in the 'where' clause must also map to a real column in the table.
 - If an element to be masked maps to an XML column in the table as a single element, the element in the 'where' clause must map to the same XML column as a single element.
- **Multiple feature option entries for the same field.** It's possible that different schemas in the system have a similar type of data that may be masked based on different conditions. For example, imagine that an implementation has different schemas that captured or referenced person identifiers in different ways:
 - One schema captures a single person ID without any corresponding "type" record and it should always be masked using Algorithm CM_SSN_MASK:

```
<personSSN mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
```

- One schema captures a person ID and a corresponding ID Type and it should be masked with Algorithm CM_SSN_MASK if the type is "SSN" and masked with algorithm CM_FEIN_MASK if the type is "FEIN".

```
<personIdType mapXML=BO_DATA_AREA mdField=ID_TYPE_CD/>  
<personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
```

- One schema captures a person ID and a corresponding ID Type and it has the same masking rules as the previous schema, but a different field name is used for the ID Type code. (This scenario could happen if for example a different label is desired for ID Type on the user interface for this schema.)

```
<personIdType mapXML=BO_DATA_AREA mdField=CM_ID_TYPE/>  
<personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
```

For this scenario, the feature options may look like this:

1. **field="PER_ID_NBR", alg="CM_SSN_MASK"**
2. **field="PER_ID_NBR", alg="CM_SSN_MASK", where="ID_TYPE_CD='SSN'"**
3. **field="PER_ID_NBR", alg="CM_FEIN_MASK", where="ID_TYPE_CD='FEIN'"**
4. **field="PER_ID_NBR", alg="CM_SSN_MASK", where="CM_ID_TYPE='SSN'"**
5. **field="PER_ID_NBR", alg="CM_FEIN_MASK", where="CM_ID_TYPE='FEIN'"**

For each schema, the system will first find whether the element applies to any masking option. It will find 5 masking options for the field PER_ID_NBR. Then it will determine if any sibling elements match the 'where' clause.

- If more than one sibling element matches a 'where' clause, a runtime error is issued. For example if a schema has an element that references "mdField=ID_TYPE_CD" and an element that references "mdField=CM_ID_TYPE", this is an error. Additionally, if multiple elements reference mdField=ID_TYPE_CD, this is an error.
- If one and only one sibling element matches a 'where' clause, the value of the element is compared to the values defined in the 'where' clause. If it finds a match on the value, the appropriate masking algorithm is applied. If no match is found (for example, the Person ID Type is "LICENSE") the element is displayed as is.
- If no sibling element matches a 'where' clause and a feature option exists with no 'where' clause (option 1 above), then the masking algorithm of the option with no 'where' clause is applied.
- **Changing the value in the 'where' clause.** If your implementation has some users that are allowed to change records where some data is masked based on a condition, it is recommended to design the user interface to reset the masked value when the value in the 'where' clause changes. For example, if a user is prevented from viewing a person's social security number, but the user is allowed to make updates to the person's record, changing the value of the Person ID Type should reset the Person ID Number. This would ensure that the user does not 'unmask' the social security number by simply changing the ID Type.

Records Maintained Using Page Maintenance

For data that is accessed via a page maintenance service call, indicate the table name and the field name where the data resides: **table="table_name", field="fld_name", alg="algorithm name"**

For example if the Person record and its collection of identifiers are displayed and maintained using page maintenance, the option value should be **table="CI_PER_ID", field="PER_ID_NBR", alg="algorithm name"**

A "where" clause may also be specified: **table="table_name", field="fld_name", where="fld_name='value'", alg="algorithm name"**

This is useful for data that resides in a child table where only data of a certain type needs to be masked. For the person ID example, **table="CI_PER_ID", field="PER_ID_NBR", alg="algorithm name", where="ID_TYPE_CD='SSN'"**

Characteristic Data

For data that is stored as a characteristic, simply indicate the characteristic type: **CHAR_TYPE_CD='char type', alg="algorithm name"**

This needs to be defined only once regardless of which characteristic entity the char type may reside on. Note that only ad-hoc characteristics are supported.

Masking Fields in Explorer Zones or Info Strings

In explorer zones data is often retrieved using SQL directly from the database. No masking is applied automatically in this case. If there is data in the explorer zone results that should be masked, the masking must be applied by calling a business service.

Similarly, an MO Info algorithm may not use BO interaction to get data. It may access data using SQL for efficiency purposes. No masking is applied when retrieving data via SQL. To apply masking to a string prior to including it in an info string, the masking must be applied by calling a business service.

The system supplies two business services to be called to determine if masking rules apply for a specific field.

- **F1-TableFieldMask** - Mask a Table field. This business service receives a table name, field name and one or more field values. If masking applies it returns the masked value.
- **F1-SchemaFieldMask** - Mask a Schema field. This business service receives a schema name and type, XPath and field value. If masking applies it returns the masked value.

Search Service Results

For data that is displayed on a 'fixed' search page, it is retrieved via a search service call. Indicate the search name and the appropriate field to mask along with the masking algorithm. For example: **search="SearchServiceName", field="PER_ID_NBR", where="ID_TYPE_CD='SSN'", alg="algorithm name"**

To find the name of the search service, launch the search in question, right click in the filter area and choose View Source. Search for ServiceName. The service name is listed there. To find the field name to mask, go back to the search window and right click on the results area and choose View Source. Look for the Widget Info section and find the field name in the SEARCH RESULTS (do not include the \$). Note, the "where" statement can only apply to fields that are also part of the search results.

Additional Masking Information

The following points provide additional information to assist in your masking configuration:

- If the demonstration database includes a **Data Masking** feature configuration, review the settings because it will probably contain masking rules that will match your own.
- On data input pages, a user might be able to enter or change masked data, such as a bank account number, but not be able to subsequently see what they added or changed.
- External systems can request information by performing a service call via a web service. Please keep in mind that some web service requests require data to be masked and some do not. For example, a request from an external system to synchronize person information needs the person's social security number unmasked; whereas a request from a web self service application to retrieve the same person information for display purposes needs the person's social security number masked. To implement this type of requirement, different users must be associated with each of the requests and these users must belong to separate user groups with different access rights.
- If a maintenance object (MO) contains a field that holds an XML document and a service call invokes the MO's service program directly, the system will mask individual XML elements in the field if a **Determine BO** algorithm has been plugged into the [maintenance object](#) and the element(s) in the respective BO schema have been secured as described above.

Database Encryption and Masking

The functionality described in this section is used to encrypt data when storing it in the database. This functionality is mutually exclusive from the User Interface Masking functionality described in the previous section. The following points highlight the features of the encryption functionality:

- The encryption key is defined using a keystore, which must be set up in order to use this functionality. For details about setting up the keystore in the system, see the Installation Guide.
- When a field is configured to be encrypted, the encrypted data is stored in a special encryption field that is not the source field (the one exposed to the user on the user interface). The source field captures the data as masked. Because a special field is required to support encryption, the product must provide support for that field to be encrypted.
- For encrypted data that must allow searching, the system supports capturing a hash value in a special field. The product must provide support for this functionality. Besides providing a special field to capture the hash value, base search functionality for that data must also cater for this configuration.
- The system supports encrypting data that is captured as an element within an XML field. If the XML field is provided in a schema owned by the product, then the product must provide specific support for the capture of the encrypted data.

The following sections provide additional information about the support for encryption provided by the framework. Refer to the security chapter of the administration guide for your particular product for more information.

Encrypting and Masking the Data

When a product enables encrypting for a given type of data, a special encryption field should be created to capture the encrypted value. Because encrypting is optional, the source field (the one exposed to the user) should not be this special encrypted field. If encryption is configured, the system will internally populate the encrypted field. The source field will be populated with asterisks by default. That way the masked data is what is shown to the user on page rather than the encrypted value.

The following points highlight how the system behaves when encryption is configured and when it is not. Assume as an example, the field is a credit card number. The user views and populates a field with the field name `CC_NBR`. The table also has a second field `ENCR_CC_NBR`. A user populates the credit card number:

- If encryption is not configured, `CC_NBR` will be updated with the entered credit card number and `ENCR_CC_NBR` will be empty. Note that in this case, an implementation may choose to configure [user interface masking](#).
- If encryption is configured, `CC_NBR` will be updated with `'*****'` and `ENCR_CC_NBR` will contain the encrypted value. The asterisks for the standard field will fill the full field size up to 50 characters.

If for some reason the standard masking using all asterisks is not desired, the system supports supplying an explicit masking algorithm using the same [Data Masking](#) plug-in spot used for [User Interface Masking](#).

WARNING: Unlike user interface masking, the masking of encrypted fields is not driven by security. The data stored in the source field for all encrypted data should be masked. Be sure not to configure security authorization logic in algorithms used for this type of masking.

Feature Option Configuration

Create a feature configuration with a Feature Type of **Encryption**. For each source field you are encrypting, enter an option with option type of **Field Encryption**. The value will contain mnemonics that reference the appropriate encryption key alias defined in the keystore along with configuration related to the field and its table location. Unlike the user interface data masking, the configuration for data encryption is related to how the data is stored rather than how it is displayed. In addition, each entry may define an explicit masking algorithm to override the default and if supported, may also define a hash field and hash alias.

For data that is stored in a specific column on a table, an explicit field to capture the encrypted value must exist. Indicate the table name, source field name and encrypted field name along with the alias: **table='table_name', field='fld_name', encryptedField='encr_fld_name', alias='alias key'**

A "where" clause may also be specified when data resides in a child table and only data of a certain type needs to be encrypted.

Example, **table='CI_PER_ID', field='PER_ID_NBR', encryptedField='ENCR_PER_ID_NBR', alias='key alias', where='ID_TYPE_CD='SSN''**

For data that is stored in an XML column in a record, the source field to be encrypted must reference a meta-data field name in its schema definition along with the element that captures the encrypted data and the alias: **field='field_name', encryptedField='encr_field_name', alias='key alias'**

The syntax for adding a reference to a masking algorithm is **maskAlg='algorithm name'** .

The syntax for adding configuration for capturing a hash value for searching purposes is **hashAlias='hashAliasKey' hashField='HASH_FLD_NAME'**.

The following is an example of configuration that uses all the possible options (specific masking algorithm, where clause and hash field support):

table='CI_PER_ID', field='PER_ID_NBR', alias='aliasKey', encryptedField='ENCR_PER_ID_NBR', hashAlias='hashAliasKey' hashField='HASH_PER_ID_NBR', where='ID_TYPE_CD=SSN', maskAlg='CM-PERIDMASK'

Searching by an Encrypted Value

If the product supports a hashed value for an encrypted field for searching purposes, the following points highlight explorer zone configuration for this purpose

- The user filter value should reference the source field and should include an additional **encrypt=** mnemonic. For example

```
type=STRING
label=PER_ID_NBR
encrypt=[CI_PER_ID,PER_ID_NBR,ID_TYPE_CD,F1]
```

Refer to [User Filters](#) for more information.

- The SQL should include the hashed value in the WHERE clause. Note that because encryption is optional, a product zone that includes searching by a field eligible for encryption will include finding a match for the filter in the source field (as plain text) or in the hashed field. For example:

```
WHERE
  [(F2) (ID.PER_ID_NBR =:F2 OR ID.HASH_PER_ID_NBR = :F2)]
```

Customizing Encryption Algorithm

Although the encryption algorithm to use with a given key can be gleaned from the key in the keystore, there is sometimes extra information associated with an algorithm that might need to be used to encrypt or decrypt data.

The system provides a feature configuration option for the **Encryption** feature type using the option type **Algorithm Info** that can be used to adjust the behavior of the encryption.

- You can modify the default mode and padding of the encryption algorithm.
- If a key will be used to digitally sign anything, the signing algorithm can also be specified for the key.

For details about the syntax, refer to the feature option type's detailed description.

The Base Package Controls One User, One User Group, And Many Application Services

When the system is initially installed, the following information is delivered:

- Application services for all secured transactions, maintenance objects, business objects, business services, scripts and zones in the base package.
- A user identified by the user id **SYSUSER**.
- A user group identified by the user group code **ALL_SERVICES**. This user group is associated with all supported application services delivered with the base product. This user group is given access to all access modes for all application services (i.e., all actions on all transactions).
- The user **SYSUSER** is linked to the **ALL_SERVICES** user group. This means that this user has access to all transactions and all actions.

You cannot change or remove the information delivered for **ALL_SERVICES**. This information is owned by the base package. It is provided so that an "initial user" has access to the entire system and can setup user groups and users as per your organization's business requirements. It is not recommended to provide your own users with access to the **ALL_SERVICES** user group. Rather, create user groups that are appropriate for the organization's business requirements and define user access to these user groups. If you introduce new transactions, configure them for the appropriate custom user groups.

In addition, **SYSUSER** is provided to allow for an initial user to define appropriate users in your implementation. Once proper administrative users have been defined, it is recommended that **SYSUSER** is updated to set the User Enable setting to Disabled.

When you receive an upgrade:

- New application services are delivered for the new transactions, business objects, zones introduced in the release. The release notes highlights the additions / changes.
- Existing application services are updated with changes in their access modes (e.g., if a new action is added to a transaction, its application service is updated accordingly).
- The **ALL_SERVICES** user group is updated so it can access the new / changed application services.
- Implementations should review the release notes and determine which user groups created for your implementation should be updated with the additions, if applicable.

Importing Security Configuration from an External Source

The product provides support for importing security information from an external source:

- If your organization uses Lightweight Directory Access Protocol (LDAP), you can import your existing LDAP users and groups into the system. Once imported, all user and group functions are available. You can import a user group, or a single user. You can resynchronize your LDAP users and groups at any time.

FASTPATH: For more information refer to [LDAP Integration](#).

- The system provides an integration with Oracle Identity Manager. When a user is created in the identity manager product, its information can automatically be interfaced to the product. Once the user is successfully created in the system, all functions are available.

FASTPATH: For more information refer to [Oracle Identity Manager Integration](#).

The Big Picture of Row Security

Some products allow you to limit a user's access to specific rows. For example, in Oracle Utilities Customer Care and Billing, row level security prevents users without appropriate rights from accessing specific accounts.

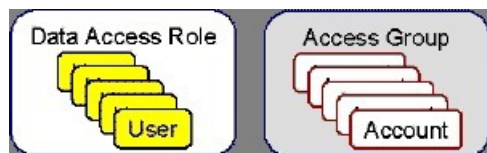
By granting a user access rights to an account, you are actually granting the user access rights to the account's bills, payment, adjustments, orders, etc.

The topics in this section describe basic row level security objects.

FASTPATH: Refer to your product's documentation for more information on row level security, if applicable.

Access Groups, Data Access Roles and Users

We'll use an example from Oracle Utilities Customer Care and Billing to describe how access groups and roles are used to restrict access to accounts. The following diagram illustrates the objects involved with account security:



An **Access Group** defines a group of **Accounts** that have the same type of security restrictions. A **Data Access Role** defines a group of **Users** that have the same access rights (in respect of access to accounts). When you grant a data access role rights to an access group, you are giving all users in the data access role rights to all accounts in the access group.

The following points summarize the data relationships involved with account security:

- An account references a single **access group**. An **access group** may be linked to an unlimited number of **accounts**.
- A **data access role** has one or more **users** associated with it. A **user** may belong to many **data access roles**.
- A **data access role** may be linked to one or more **access group**. An **access group** may be linked to one or more **data access roles**.

If you use row level security, setting up your access roles and groups can be easy or challenging - it all depends on your organization's requirements. Refer to the product's **Administration Guide - Implementing Account Security** for several case studies. These case studies provide examples of how different requirements can be implemented using these concepts.

Defining Application Services

An application service exists for every transaction in the system. Please refer to [Application Security](#) for a description of how application services are used when you grant user groups access rights transactions.

CAUTION: Important! If you introduce a new application service, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Application Service - Main

Select **Admin > Security > Application Service** to define an application service.

Description of Page

Enter a unique **Application Service** code and **Description** for the application service.

Indicate the application service's various **Access Modes** (i.e., actions). Refer to [Action Level Security](#) for more information about the significance of these fields.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [SC_APP_SERVICE](#).

Application Service - Application Security

Use the Application Security portal to set up security for an application service.

Open this page using **Admin > Security > Application Service**, and then navigate to the **Application Security** tab.

This section describes the available zones on this page.

Application Service Details zone. This zone contains display-only information about the selected application service, including the Access Modes for the application service and its security type.

User Groups Linked zone. This zone lists the user groups that currently have a link to the application service. Note that expired links are also included. The following actions are available:

- Click the **Description** link to navigate to the [User Group - Users](#) page for the adjacent user group. This allows you to add or remove users linked to the user group.
- Click **Deny Access** to remove the selected Application Service's link to this user group.

User Groups not Linked zone. This zone lists the user groups that do not have a link to the application service. The following actions are available:

- Click the **Description** link to navigate to the [User Group - Users](#) page for the adjacent user group.
- Click **Grant Access** to navigate to the [User Group - Application Services](#) page for the user group. The page is automatically positioned at the selected application service allowing you to set the access modes and the expiration date.

Defining Security Types

Security types are used to define the types of [field level security](#).

NOTE: Programming is required. You cannot have field level security without introducing logic to user exits. Refer to [Field Level Security](#) for more information on how security types are used to define field level security.

Security Type - Main

Select **Admin > Security > Security Type** to define your security types.

Description of Page

Enter a unique **Security Type** and **Description**.

Use the **Authorization Level** grid to define the different authorization levels recognized for this security type. Enter an **Authorization Level Number** and its **Description**.

NOTE: Programming is required. Note that the values that you enter are not interpreted by the system itself, but by the user exit code used to implement the special security. Check with the developer of the user exit logic for the correct values. Refer to [Field Level Security](#) for more information on how security types are used to define field level security.

Use the **Application Services** grid to define the application service(s) to which this security type is applicable. If this application service is already associated with user groups, you must update each user group to define their respective security level. This is performed using [User Group - Application Service](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SC_TYPE](#).

Defining User Groups

A user group is a group of users who have the same degree of security access. Think of a user group as a "role"; associated with a role are:

- The users who play this role
- The application services to which the role's users have access (along with the actions they can execute for each service and their field level security authorization levels).

User Group - Main

Select **Admin > Security > User Group** to view the application services to which a user has access.

CAUTION: Application services may not be changed or removed from the **ALL_SERVICES** user group. Refer to [The Base Package Controls One User, One User Group, And Many Application Services](#) for an explanation.

Description of Page

Enter a unique **User Group** code and **Description** for the user group.

Owner indicates if this user group is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a user group. This information is display-only.

The **Application Services** grid displays the various application services to which users in this group have access. Please note the following in respect of this grid:

- Use the **Application Service** search to restrict the application services displayed in the grid. For example, if you only want to see application services that start with the word "field", you can enter this word and press enter.
- To add additional application services to this user group, navigate to the [User Group - Application Services](#) page and click the add icon.
- To remove or change this user group's access to an application service, click the go to button adjacent to the respective application service. This will cause you to be transferred to the [User Group - Application Services](#) tab where you should click the - icon to remove the application service from the user group.
- Note, **Owner** indicates if this user group / application service relationship is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an application service to the user group. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [SC_USER_GROUP](#).

User Group - Application Services

Select **Admin > Security > User Group** and navigate to the **Application Services** tab to maintain a user group's access rights to an application service.

NOTE: Important! When you grant a user group access rights to an application service, you are actually granting all users in the user group access rights to the application service.

Description of Page

The **Application Service** scroll contains the application services to which the **User Group** has access.

NOTE: You can also use Main page to select the application service for which you wish to change the access privileges. To do this, simply click the go to button adjacent to the respective application service.

To add additional application services to this user group, click the + icon and specify the following:

- Enter the **Application Service ID** to which the group has access.
- Define the **Expiration Date** when the group's access to the application service expires.

Define the **Access Modes** that users in this group have to the **Application Service**. When a new application service is added, the system will default all potential **Access Modes** associate with the **Application Service**. You need only remove those modes that are not relevant for the **User Group**. Refer to [Action Level Security](#) for more information about access modes.

CAUTION: Important! If an application service supports actions that modify the database other than **Add, Change,** and **Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a transaction that supports actions in addition to **Add, Change,** and **Inquire** (e.g., **Freeze, Complete, Cancel**). If you want to give a user access to any of these additional actions, you must also give the user access to the **Inquire** and **Change** actions.

If you require additional security options, often referred to as "field level" security, then you use **Security Type Code** and assign an **Authorization Level** to each. When a new application service is added, the system will display a message indicating how many security types are associated with this application service. Use the search to define each Security Type Code and indicate the appropriate Authorization Level for this user group. Refer to [Field Level Security](#) for more information about security types.

User Group - Users

Select **Admin > Security > User Group** and navigate to the **Users** tab to maintain the users in a user group.

Description of Page

The scroll area contains the users who are part of this user group.

NOTE: Keep in mind that when you add a **User** to a **User Group**, you are granting this user access to all of the application services defined on the **Application Services** tab.

The following fields are included for each user:

- Enter the **User ID** of the user.
- Use **Expiration Date** to define when the user's membership in the group expires.
- **Owner** will be **Customer Modification**.

NOTE: You can also add a user to a user group using [User - Main](#).

Defining Access Groups

FASTPATH: Refer to [The Big Picture of Row Security](#) for a description of how access groups are use to restrict access to specific objects.

Access groups control which groups of users (referred to as Data Access Roles) have rights to accounts (or other objects) associated with the access group. Select **Admin > Security > Access Group** to define your access groups.

Description of Page

Enter a unique **Access Group** code and **Description** for the data access group.

Use the **Data Access Role** collection to define the data access roles whose users have access to the access group's accounts (or other objects). Keep in mind that when you add a **Data Access Role** to an **Access Group**, you are granting all users who belong to this role access to all of the accounts (or other objects) linked to the access groups. Refer to [Access Groups, Data Access Roles and Users](#) for more information.

NOTE: You can also use [Data Access Role - Access Group](#) to maintain a data access role's access groups.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ACC_GRP](#).

Defining Data Access Roles

FASTPATH: Refer to [The Big Picture of Row Security](#) for a description of how access groups are use to restrict access to specific objects.

The data access role transaction is used to define two things:

- The users who belong to the data access role.
- The access groups whose accounts (or other objects) may be accessed by these users.


Data Access Role - Main

Select **Admin > Security > Data Access Role** to define the users who belong to a data access role.

Description of Page

Enter a unique **Data Access Role** code and **Description** for the data access role.

The scroll area contains the **Users** who belong to this role. A user's data access roles play a part in determining the accounts (or other objects) whose data they can access. Refer to [Access Groups, Data Access Roles and Users](#) for more information.

To add additional users to this data access role, press the add button, , and specify the following:

- Enter the **User ID**. Keep in mind that when you add a **User** to a **Data Access Role**, you are granting this user access to all of the accounts (or other objects) linked to the data access role's access groups.
- Use **Expiration Date** to define when the user's membership in this data access role expires.

NOTE: Also maintained on the user page. You can also use [User - Access Security](#) to maintain a user's membership in data access roles.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DAR](#).

Data Access Role - Access Group

Select **Admin > Security > Data Access Role** and navigate to the **Access Groups** tab to define the access groups whose accounts (or other objects) may be accessed by the users in this data access role.

Description of Page

Use the **Access Group** collection to define the access groups whose objects can be accessed by this role's users. Keep in mind that when you add an **Access Group** to a **Data Access Role**, you are granting all users who belong to this role access to all of the accounts (or other objects) linked to the access groups. Refer to [Access Groups, Data Access Roles and Users](#) for more information.

NOTE: You can also use [Access Group - Main](#) to maintain an access group's data access roles.

Defining Users

The user maintenance transaction is used to define a user's user groups, data access roles, portal preferences, default values, and To Do roles. To access the user maintenance transaction, select **Admin > Security > User**.

The user maintenance transaction is the same transaction invoked when the user launches [Preferences](#).

User Interface Tools

This section describes tools that impact many aspects of the user interface.

Defining Menu Options

The contents of this section describe how you can add and change menus.

CAUTION: Updating menus requires technical knowledge of the system. This is an implementation and delivery issue and should not be attempted if you do not have previous experience with menus.

NOTE: Security and menus. Refer to [Application Security](#) for a discussion of how application security can prevent menu items (or an entire menu) from appearing.

NOTE: Module configuration and menus. Your [module configuration](#) can prevent menu items (or an entire menu) from appearing.

Menu - Main

This transaction is used to define / change any menu in the system. Navigate to this page using **Admin > System > Menu**.

Description of Page

Enter a meaningful, unique **Menu Name**.

Owner indicates if this menu line is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a menu line. This information is display-only.

The **Flush Menu** button is used to flush the cached menu items so you can see any modified or newly created menus. Refer to [Caching Overview](#) for more information.

Menu Type defines how the menu is used. You have the following options:

- **Admin** is one of the menus that appears in the Application Toolbar. It is a special type of menu because [admin menu](#) items can be grouped alphabetically or by functional group. Refer to the description of Admin Menu Order on [Installation Options - Framework](#) for more information about admin menu options.
- **Context** refers to a [context menu](#).
- **Main** is another menu that appears in the Application Toolbar that is simply titled [Menu](#).
- **Page Action Menu** defines buttons that appear in the [Page Title Area](#).
- **Submenu** defines a menu group that appears when an Application Toolbar menu is selected. for the Admin menu, this is only visible when it's organized functionally.
- Enter **User Menu** refers to the menu items that appear on the [user menu](#); for example, User Preferences.

Description provides a description of the menu. Note that this is not the text used when displaying a menu option.

Sequence is only enabled for the **Main** and **Admin** menu types.

The grid contains a summary of the menu's lines. Besides the standard add and delete icons available in a grid, the following information is displayed:

- **Menu Line ID** is the unique identifier of the line on the menu. This information is display-only. Before the menu line id is a Go To icon that allows a user to drill into the Menu Items for the displayed menu line.
- **Sequence** is the relative position of the line on the menu. Note, if two lines have the same **Sequence**, the system organizes the lines alphabetically (based on the **Long Label**, which is defined on the next tab).

NOTE: An implementation may override the sequence of a base product owned menu line. Also note that the sequence is defined on the menu line language table, allowing for different orders to be used for different languages (or to let the menu be sorted alphabetically in one language and in a specified order in a different one).

- **Navigation Option / Submenu** contains information about the line's items. If the line's item invokes a submenu, the submenu's unique identifier is displayed. If the line's item(s) invoke a transaction, the description of the first item's [navigation option](#) is displayed.

- **Long Label** is the verbiage that appears on the menu line.
- **Item Count** is the number of menu items on the line.
- **Owner** indicates if this menu line is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a menu line. This information is display-only.

NOTE: Adding menu lines to base owned menus. An implementation may choose to add custom menu lines along with its menu item (or items) to a base owned menu.

Refer to the description of [Menu Items](#) for how to add items to a menu line.

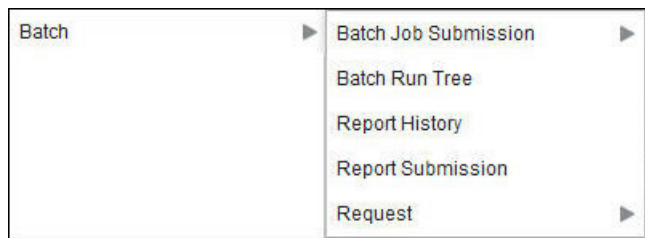
Menu - Menu Items

Once a menu has lines (these are maintained on the main page), you use this page to maintain a menu line's items.

Each menu line can contain one or two menu items. The line's items control what happens when a user selects an option on the menu.

There are two types of menu lines that define a single menu item: one type causes a submenu to appear; the other type causes a transaction or script to be invoked when it's selected.

- The following is an example of a menu line with a single item that opens a submenu:



- The following is an example of a menu line with a single menu item that launches a transaction or script:



A menu line that defines two menu items is used to provide an Add option and a Search option for the same type of object. In this case each menu item defines a transaction or script to be launched. The menu is rendered with the Add and Search options displayed. The following is an example of a menu line with two menu items.



Navigate to this tab by clicking the Go To button adjacent to a menu line from the Main tab.

Description of Page

Menu Name is the name of the menu on which the line appears. **Menu Line ID** is the unique identifier of the line on the menu. **Owner** indicates if this menu is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

The **Menu Line Items** scroll contains the line's menu items. The following points describe how to maintain a line's items:

- **Menu Item ID** is the system assigned unique identifier of the item.
- **Owner** indicates if this item is owned by the base product or by your implementation (**Customer Modification**).
- If the menu item should invoke a submenu:
 - Use **Sub-menu Name** to identify the menu that should appear when the line is selected

- Use **Long Label** to define the verbiage that should appear on the menu line
- Populate the **Override Label** to override the long label of a base product owned sub-menu.
- If the item should invoke a transaction or BPA script:
 - Use **Sequence** to define the order the item should appear in the menu line (we recommend this be set to **1** or **2** as a menu line can have a maximum of 2 menu items). The “search” menu item should be defined as sequence 1 and the “add” menu item as sequence 2 given that the label of the “search” menu item is used for the menu line’s label.
 - Use **Navigation Option** to define the transaction or script to open. Refer to [Defining Navigation Options](#) for more information.
 - For a menu line that includes two items — one for Add and one for Search, if one of the items includes configuration for **Image GIF Location and Name** , the system assumes that this represents the Add. This functionality is a carry over from earlier releases where the Add function rendered in the menu with a “+” image, which also identified the item that represents the Add. If neither item includes Image configuration (because it is no longer needed for rendering the menu), the system relies on the order of the items as mentioned above. The first item is the “search” and the second item is the “add”.
 - **Image Height, Image Width and Balloon Description** are not applicable at this time.
- Use the **Long Label** to define the text to appear on the menu entry. Note that when a menu line defines two menu items, the long label on the search entry is used to build the menu entry text. The label long on the menu line that defines the Add option is information only.
- The **Override Label** is provided in case you want to override the base-package's label.
- Use **Application Service** and **Access Mode** to easily suppress a menu item for one or more users. Refer to [Application Security](#) for more information.

The Big Picture of System Messages

All error, warning and informational messages that are displayed in the system are maintained on the message table. Every message is identified by a combination of two fields:

- **Message category number.** Think of a message category as a library of messages related to a given functional area. For example, there is a message category for billing messages and another one for payment messages.
- **Message number.** A unique number identifies each message within a category.

Every message has two components: a brief text message and a long description. On the **Main** tab, you can only maintain the brief message. If you need to update a message's long description, you must display the message on the **Details** tab.

NOTE: You cannot change the product’s text. If the message is "owned" by the product, you cannot change the product’s message or detailed description. If you want your users to see a different message or detailed description other than that supplied by the product, display the message on the **Details** tab and enter your desired verbiage in the "customer specific" fields (and flush the [cache](#)).

Defining System Messages

The contents of this section describe how to maintain messages that appear throughout the system. An implementation may introduce messages used in custom processes or may choose to override the text for messages delivered by the product.

Message - Main

Select **Admin > System > Message** to maintain a message category and its messages.

Description of Page

To add a new message category, enter a **Message Category** number and **Description**.

CAUTION: Message category 80000 or greater must be used to define new messages introduced for a specific implementation of the system. Changes to other Message Text will be overwritten when you next upgrade. If you want to make a change to a Message, drill down on the message and specify Customer Specific Message Text. Note that even for message categories 80000 and higher, message numbers lower than 1000 are reserved for common base product messages.

NOTE: Owner indicates if this message category is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a category. This information is display-only.

To update a message, you must first display its **Message Category**. You can optionally start the message grid at a **Starting Message Number**.

To override the message text or detailed description of messages owned by the base product, click on the message's go to button. When clicked, the system takes you to the **Details** tab on which you can enter your implementation's override text.

The following points describe how to maintain messages owned by your implementation:

- Click the - button to delete a message.
- Click the + button to add a new message. After clicking this button, enter the following fields:
- Use **Message Number** to define the unique identifier of the message within the category.
- Use **Message Text** to define a basic message. You can use the %n notation within the message text to cause field values to be substituted into a message. For example, the message text **The %1 non-cash deposit for %2 expires on %3** will have the values of 3 fields merged into it before it is displayed to the user (%1 is the type of non-cash deposit, %2 is the name of the customer, and %3 is the expiration date of the non-cash deposit).

NOTE: The system merges whatever values are supplied to it. Therefore, if a programmer supplies a premise address as the second merge parameter in the above message, this address is merged into the message (rather than the customer's name).

- **Owner** indicates if this message number is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a message. This information is display-only.
- Click the go to button to enter a detailed description of the message. Clicking this button transfers you to the **Details** tab.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MSG](#). In addition, messages are used throughout the system for error messages and other system messages.

Message - Details

Select **Admin > System > Message** and navigate to the **Details** tab to define detailed information about a message.

NOTE: Drilling in from the Main tab. Rather than scrolling through the messages, you can display a message by clicking the respective go to button in the grid on the main tab.

Description of Page

The **Message Collection** scroll contains an entry for every message in the grid on the Main tab. It's helpful to categorize messages into two categories when describing the fields on this page:

- Product messages

- Implementation-specific messages (i.e., a message added to **Message Category** 80000 or greater)

For product messages, you can use this page as follows:

- If you want to override a message, specify **Customer Specific Message Text**.
- You are limited to the same substitution values used in the original **Message Text**. For example, if the original **Message Text** is **The %1 non-cash deposit for %2 expires on %3** and %1 is the type of non-cash deposit, %2 is the name of the customer, and %3 is the expiration date of the non-cash deposit; your **Customer Specific Message Text** is limited to the same three substitution variables. However, you don't have to use any substitution variable in your message and you can use the substitution variables in whatever order you please (e.g., %3 can be referenced before %1, and %2 can be left out altogether).
- If you want to override the detailed description of an error message, specify **Customer Specific Description**. Note that the system does not present detailed descriptions when warnings are shown to users. Therefore, it doesn't make sense to enter this information for a warning message.

For implementation-specific messages, you can use this page as follows:

- **Message Text** is the same Message Text displayed on the main tab.

CAUTION: If both **Message Text** and **Customer Specific Message Text** are specified, the system will only display the **Customer Specific Message Text** in the dialog presented to the user.

- Use **Detailed Description** to define additional information about an error message. Note that the system does not present detailed descriptions when warnings are shown to users. Therefore, it doesn't make sense to enter this information for a warning message.

CAUTION: If both **Detailed Description** and **Customer Specific Description** are specified, the system will only display the **Customer Specific Description** in the dialog presented to the user.

The Big Picture of Portals and Zones

A portal is a page that is comprised of one or more information zones. The base product pages are built using either a fixed page metaphor or using portals and zones. The contents of this section describe general information about portals and zones.

There Are Three Types of Portals

There are three broad classes of portals:

- **Standalone Portal.** Standalone portals are separate pages where the main tab of the page is built using a portal. These pages are opened using any of the standard methods (e.g., by selecting a menu item, by selecting a favorite link, etc.). Additional tabs for a stand-alone portal may be included using tab page portals.
- **Tab Page Portals.** These types of portals cannot be attached to a menu. They simply define the zones for a tab on either a standalone portal or on a “fixed” page. Please contact customer support if you need to add portals to existing transactions.
- **Dashboard Portal.** The dashboard portal is a portal that appears in the [Dashboard Area](#) on the user's desktop. Its zones contain tools and information that exist on the user's desktop regardless of the transaction.

There is only one dashboard portal. This portal and several zones are delivered as part of the base-package. Your implementation can add additional zones to this portal. Please contact customer support if you need to add zones to the dashboard portal.

Common Characteristics of All Portals

The topics that follow describe characteristics common to all types of portals.

Portals Are Made Up of Zones

A portal is a page that contains one or more zones, and each zone contains data of some sort.

All zones reference a **Zone Type**. The zone type controls the behavior of the zone and the parameters available to configure the zone.

Configuring Zones for a Portal

The portal includes configuration of how the zones should appear on the portal by default. This includes the following options, all of which may be overridden by an implementation.

- The order in which the zone should appear. An implementation may configure an override sequence to change the order zones on a base delivered portal.
- Whether the zone is visible on the portal. Zones delivered in the base product should be configured to be visible. But an implementation may override this if desired.
- Whether the zone should display initially collapsed or not. A zone's data is only retrieved when it is expanded. As such, a zone may be configured to be initially collapsed when the data is not needed very often. A user can expand the zone when the information is required. Implementations may change the collapsed setting of a base product portal / zone.

FASTPATH: Refer to [Zones May Appear Initially Collapsed When a Page Opens](#) for more information.

FASTPATH: Refer to [Defining Portals](#) for more information about this configuration.

In addition, the portal includes configuration to indicate whether or not the portal should appear on a user's portal preferences. This is typically enabled for a portal that provides disparate information where not all zones are applicable to all users or where users may wish to adjust the order of the zones. An example of a portal enabled for portal preferences is the Dashboard portal. The user can override zone oriented configuration for the portal:

- Which zones appear on that portal
- The order in which the zones appear
- Whether the zones should be initially collapsed when the portal opens.
- The refresh seconds. This is applicable to zones displaying data that changes often.

An implementation can optionally configure the system to define portal preferences on one or more "template" users. When a template user is linked to a "real" user, the real user's preferences are inherited from the "template" user and the "real" user cannot change their preferences. Some implementations opt to work this way to enforce a standard look and feel for users in the same business area.

FASTPATH: Refer to [User — Portal Preferences](#) for more information about how users configure their zones.

Granting Access to Zones

An [application service](#) is associated with each zone. A user must be granted access rights to the respective application service in order to see a zone on a portal.

FASTPATH: Refer to [The Big Picture Of Application Security](#) for information about granting users access rights to an application service.

Please note the following with respect to zone application security:

- For most base product portals, all the zones for all the portals reference the same application service that is used to grant access to the main (stand-alone) portal for the page. In other words, if the user has access to the page, then he has access to all the zones on all portals for the page. There may be exceptions to this rule for certain portals.
- For a base product multi-query zones, typically the individual query zones and the multi-query zone reference the same application service that is used to grant access to the main (stand-alone) portal for the page. However, there may be individual query zones provided with a unique application service. This may occur when the query option is unusual and not applicable to all users or even to all implementations. If a user does not have security access to an individual query zone, that option will not be available in the dropdown.
- For base product portals that are configured to show on portal preferences, it is common that the portal contains different types of zones that may be applicable to different types of users. Typically these types of portals will deliver a unique application service for each zone so that an implementation may configure which user groups are allowed to view each zone. For these types of portals, please note the following:
 - A user's [Portal Preferences](#) page contains a row for a zone regardless of whether the user has access rights to the zone. Because of this, the system displays an indication of the user's access rights to each zone.
 - If a user's access rights to a zone are revoked, the zone will be suppressed when the user navigates to the respective portal.
 - Revoking a user's access rights does not change the user's [portal preferences](#) (i.e., a user can indicate they want to see a zone even if they don't have access to the zone - such a zone just won't appear when the respective portal appears).

NOTE: If you don't need to use zone security. When defining a zone, an application service is required. For zones that don't require special security, the product provides a "default" application service (F1-DFLT) that may be used. The expectation is that all user groups are granted access to this application service.

Common Characteristics of Stand-Alone Portals

The topics that follow describe additional characteristics specific to [stand-alone portals](#).

Putting Portals on Menus

A stand-alone portal should appear as a menu item on one of your menus. The following points provide how to do this:

- Every stand-alone portal has an associated navigation option. You can see a portal's navigation option on the [Portal - Main](#) page.
- To add a portal to a menu, you must add a [menu item](#) to the desired menu. This menu item must reference the portal's navigation option. There are two ways to add a menu item:
- If the portal's navigation option doesn't currently exist on a menu, you can press the **Add To Menu** button on the [Portal - Main](#) page. When you press this button, you will be prompted for the menu. The system will then create a menu item on this menu that references the portal's navigation option.
- You can always use the [Menu](#) page to add, change and delete menu items.

NOTE: No limit. A portal's navigation option can appear on any number of menu items (i.e., you can create several menu items that reference the same portal).

NOTE: Favorite links. Your users can set up their preferences to include the portal's navigation option on their [Favorite Links](#). This way, they can easily navigate to the portal without going through menus.

Granting Access to A Portal

An [application service](#) is associated with each [stand-alone portal](#). A user must be granted access rights to the respective application service in order to see a portal. Tab portals do not have separate security access. If a user has access to the main stand-alone portal, then the user will have security access to all its tabs. However, as mentioned in [Granting Access to Zones](#), in some scenarios, the individual zones on the portal may have different security access rights depending on the functionality.

FASTPATH: Refer to [The Big Picture Of Application Security](#) for information about granting users access rights to an application service.

NOTE: Automatically created. When you add a new stand-alone portal, the system automatically creates an application service behind the scenes. You'll need to know the name of this application service as this is what you use to grant access to the portal. The name of each stand-alone portal's application service is shown on the portal transaction.

Please note the following in respect of how application security impacts a user's portals:

- A user's [Portal Preferences](#) page only shows the portals configured to show on user preferences and where they have security access.
- The system's menus only show portals to which a user has security access.
- Users can set up favorite links to all portals, but they must have security rights to the portal's application service in order to invoke the favorite link.

Custom Look and Feel Options

The default look and feel of the application can be customized via feature configuration and cascading style sheets. The base product is provided with a [Custom Look And Feel Feature Configuration](#) type. You may want to set up a feature configuration of this type to define style sheet and UI Map help options.

User Interface

The base product allows for the conditional inclusion of custom style sheets into the system style set. Custom styles may override any style provided by the base product. The style sheet may also include new styles for use in customer zone definitions. Use the [Style Sheet](#) option on the [Custom Look And Feel Feature Configuration](#) to define your custom style sheet.

NOTE: Some styles cannot change if they are part of the HTML code.

CAUTION: Implementers must ensure that the customized user interface is stable and scalable. Changing font, alignment padding, border size, and other user interface parameters may cause presentation problems, like scrollbars appearing or disappearing, cursors not working as expected, and unanticipated look and feel alterations of some layouts.

UI Map Help

A [tool tip](#) can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map zone or UI Map. Refer to the tips context sensitive zone associated with the UI Map page for more information. The **Custom Look And Feel** Feature Configuration provides options to control the following:

- Whether UI Map Help functionality is turned on or off. By default it is turned on.
- Override the default help image with a custom image
- The location of the help image, either before or after the element.

FASTPATH: Refer to the feature configuration for a detailed description of each option.

Setting Up Portals and Zones

The topics in this section describe how to set up portals and zones. Please refer to [The Big Picture of Portals and Zones](#) for background information.

Defining Zone Types

A Zone Type represents a particular type of zone with a specific behavior. For example, a data explorer zone type is used to select data using a specific SQL statement and display the data based on parameter configuration. The zone type defines the Java Class that controls the behavior of the zone and defines the parameters that the Java Class supports. The base product supports many zone types used to build the portal / zone user interface. Implementations may introduce their own zone types.

NOTE: It is not very common for an implementation to introduce their own zone types.

Select **Admin > System > Zone Type** to maintain zone types.

Description of Page

Specify an easily recognizable **Zone Type** code and **Description**. Use the **Detailed Description** to describe in detail what the zone type does.

CAUTION: When adding new zone types, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Owner indicates if this zone type is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a zone type. This information is display-only.

Java Class Name is the Java class responsible for building the zone using the parameters defined below.

Two types of parameters are specified when defining a zone type:

- Parameter values that have a **Usage of Zone** are defined on the zones and control the functionality of each zone governed by the zone type. A **Usage** value of **Zone - Override Allowed** indicates that an implementation may override the parameter value for a base zone.
- Parameter values that have a **Usage of Zone Type** are defined directly on the zone type and control how the zone type operates (e.g., the name of the XSL template, the name of the application service). A **Usage** value of **Zone Type - Override Allowed** indicates that an implementation may override the parameter value for a base zone type.

The following points describe the fields that are defined for each parameter:

- **Sequence** defines the relative position of the parameter.

- **Parameter Name** is the name of the parameter.
- **Description** is a short description that allows you to easily identify the purpose of the parameter.
- **Comments** contain information that you must know about the parameter or its implementation. For parameters with a usage of **Zone** or **Zone — Override Allowed**, this information is visible to the user when viewing or defining this parameter for a zone of this type.
- **Usage** indicates whether the parameter value is defined in a **Zone** of this type or in the **Zone Type**. **Zone - Override Allowed** and **Zone Type - Override Allowed** indicate that override values for the parameters defined in a base zone or base zone type can be entered.
- **Required** is checked to indicate that a zone must define a value for the parameter. It is not checked if a value for the parameter is optional. This field is protected if the **Usage** is **Zone Type** or **Zone Type - Override Allowed**.
- **Parameter Value** is used to define the value of zone type parameters. This field is protected if the **Usage** is **Zone** or **Zone - Override Allowed**.
- **Owner** indicates if this parameter is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a parameter. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ZONE_HDL](#).

Zone Type Parameter Comments

For the product owned zone type parameters, the parameter's detailed description provides the detail needed for properly configuring the parameter. For the Action parameters (IMPLEMENTOR_ACTION_n), the parameter description is abbreviated. Additional detail about configuring this parameter may be found in the [Zone Action Parameter](#) detailed information. The same details apply.

Defining Zones

The contents of this section describe how to maintain zones.

Zone - Main

Implementations may use the zone page to define custom zones. In addition, an implementation may override descriptions or some parameter values for base product zones.

Select **Admin > System > Zone** to create or maintain a zone.

Description of Page

Specify an easily recognizable **Zone** identifier and **Description**. Note that if this zone appears on a portal, this description acts as the zone title.

Override Description is provided if your implementation wishes to override the description of the value provided by the product.

CAUTION: Important! When introducing a new zone, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Owner indicates if this zone is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a zone. This information is display-only.

Zone Type identifies the zone type that defines how the zone functions.

Application Service is the application service that is used to provide security for the zone. Refer to [Granting Access To Zones](#) for more information.

The **Width** defines if the zone occupies the **Full** width of the portal or only **Half**.

NOTE: Zones on the [dashboard portal](#) are always the width of the dashboard.

If the zone type supports help text, you can use **Zone Help Text** to describe the zone to the end-users. Note that for multi-query zones, if the multi-query zone has help text, that is displayed for any zone selected. If the multi-query zone does not have help text, but the selected zone has help text, the selected zone's help text is displayed. Please refer to the section on [zone help text](#) for more information on how you can use HTML and cascading style sheets to format the help text.

Use **Override Zone Help Text** to override the product provided help text for this zone.

NOTE: Viewing Your Text. You can press the **Test** button to see how the help text will look when it's displayed in the zone.

The grid contains the zone's parameter values. The Zone Type controls the list of parameters. The grid contains the following fields:

- **Description** describes the parameter. This is display-only. Note that if there is a detailed description on the zone type parameter, a question mark icon appears next to the parameter's description. Click the icon to see details related to the parameter, including tips on how to populate the parameter value.

NOTE: Additional Details for Some Parameters. There are several parameter types that have a lot of detail related to the possible configuration that cannot easily fit into the detailed description. Refer to [Common Zone Parameters for Additional Information](#) about these parameters.

- **Parameter Value** is the value for the parameter.
- Use **Override Parameter Value** to override the existing value for this parameter. This field is enabled when the related zone type parameter value is **Zone - Override Allowed**, and the zone is owned by the base product.
- **Owner** indicates if this parameter is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ZONE](#).

Zone - Portal

Select **Admin > System > Zone** and navigate to the **Portal** tab to define the portals on which a zone appears.

Description of Page

The scroll area contains the portals on which the zone appears.

To add a zone to a portal, press the + button and specify the **Portal**.

NOTE: Owner indicates if this portal / zone relationship is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

NOTE: You can also add a zone to a portal using [Portal - Main](#). Additional configuration about how the zone appears on the portal is available only on the Portal.

Zone Configuration Topics

The topics in this section provide additional information related to setting up your zones.

Zone Help Text

Most zone types support a button that allows a user to see zone-specific help text, which is defined on the zone page.

You can use HTML tags in the zone help text. The following is an example of help text that contains a variety of HTML tags:

**This zone summarizes `revenue` in 4 periods:`
`**

The above would cause the word **revenue** to be bold and blue:

- `` and `` are the HTML tags used to indicate that the surrounded text should be bold
- `` and `` are the HTML tags used to indicate that the surrounded text should be blue.

The following are other useful HTML tags:

- `
` causes a line break in a text string. If you use `

` a blank line will appear.
- `<i>` causes the surrounded text to be italicized

Please refer to an HTML reference manual or website for more examples.

You can also use "spans" to customize the look of the contents of a text string. For example, your text string could be `revenue`. This would make the word "revenue" appear as large, bold, Courier text. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

The following is an example of help text using a variety of HTML tags:

```
<font FACE="arial" size=2>
```

This zone summarizes `revenue` in 4 periods:`
`

- The `1st period` is under your control. You simply select the desired `Period`, above `<i>`(you may need to click the down arrow to expose the filter section)`</i>
`

- The `2nd period` is the period before the 1st period`
`

- The `3rd period` is the same as the 1st period, but in the previous year`
`

- The `4th period` is the period before the 3rd period`
`

```
<br>
```

The traffic light's color is determined as follows:`
`

- The ratio of the 1st and 3rd period is calculated`
`

- If this value is between 80 and 100, `yellow` is shown`
`

- If this value is < 80, `red` is shown`
`

- If this value is > 100, `green` is shown`
`

- If the value of the 3rd period is 0, no color is shown`
`

```
</font>
```

NOTE: It is possible to associate tool tip help with individual HTML and UI map elements. For more information, see [UI Map Help](#).

Zone Parameter Details

For most zone parameters, the inline help for the parameter provides the detailed information needed for configuring the parameter values. For some parameters with very detailed descriptions, the inline help is abbreviated and more detail is provided here.

SQL Statement

Data explorer zones are used to select data to display using one or more SQL statements. The SQL parameters are applicable to the following zone types

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**). The parameter has the description **SQL Statement**.
- Info Data Explorer - Multiple SQLs (**F1-DE**). The parameters follow the description pattern of **SQL Statement x**.
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**). The parameters follow the description pattern of **SQL Statement x**.

The following table provides a list of SQL substituted keywords that may be used in the SQL Statement parameters in explorer zones. At execution time, the system determines the database and substitutes the keyword with the database specific syntax:

Keyword	Description	Examples
@toCharacter()	Converts the input to Character data type.	select @toCharacter (batch_cd) as batchCode from ci_batch_ctrl
@toDate()	Converts the input to Date data type.	select @toDate (last_update_dttm) as lastUpdateDate from ci_batch_ctrl
@toNumber()	Converts the input to Number data type.	select @toNumber (next_batch_nbr) from ci_batch_ctrl
@currentDate	Fetches the current date. CAUTION: The Oracle functions SYSDATE and CURRENT_DATE should not be used because they do not properly cater for adjusting dates from the database time zone to the installation time zone, if needed.	select batch_cd, @currentDate as today from ci_batch_ctrl
@currentTimestamp	Fetches the current date / time. CAUTION: The Oracle functions SYSTIMESTAMP and CURRENT_TIMESTAMP should not be used because they do not properly cater for adjusting the date / time from the database time zone to the installation time zone, if needed.	select batch_cd from ci_batch_ctrl where last_update_dttm > @currentTimestamp
@concat	Combines the result list of two or more columns.	select batch_cd @concat next_batch_nbr concatNbr from ci_batch_ctrl
@substr(string, start)	String is the input String that you are trying to get a substring of. Start is the position of the character for the output results.	select batch_cd batchCode from ci_batch_ctrl Result: TESTCD select @substr (batch_cd,3) batchCode from ci_batch_ctrl Result: STCD
@substr(string, start, end)	String is the input String that you are trying to get a substring of. Start is the position of the character for the output results. End is the number of characters required in the output from starting position.	Select batch_cd batchCode from ci_batch_ctrl Result: TESTCD select @substr (batch_cd,3,2) batchCode from ci_batch_ctrl Result: ST
@trim	Trims the white spaces of the output on both sides.	select @trim (batch_cd) as batchCode from ci_batch_ctrl
The following syntax is related to 'fuzzy' searching. It is only applicable if Oracle DB Text is enabled and a context text index has been created. Refer to Advanced Search Options for more information.		
@fuzzy(string, score, numresult, 'weight')	String is the input value for the search. Score is the degree of 'fuzziness'. Valid values are between 1 - 80. The higher the number the more precise the search. Default is 60.	Set score to 70, number results to 6, and specify weight. select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70, 6, 'weight')) > 0

Keyword	Description	Examples
	<p>Numresults is the number of variations to consider for the string. Valid values are between 1 and 5000. Default is 100.</p> <p>Indicate 'weight' to signal that the results are returned in order of weight. Leave this setting off to indicate that the results are returned in order of score.</p>	
@fuzzy(string)	This returns a string result from the fuzzy expansion operation where the default value of 60 is assumed for the score and the default value of 100 is assumed for the numresult .	To use default values: select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1))> 0
@fuzzy(string, score)	This returns a string result from the fuzzy expansion operation with the score specified and the default value of 100 for the numresult .	Set score to 70. select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70)) > 0
@fuzzy(string, score, numresult)	This returns a string result from the fuzzy expansion operation with the similarity score and the numresults specified.	Set score to 70, number results to 6. select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70, 6)) > 0

Column Parameters

Data explorer zones are used to select data to display using one or more SQL statements. For each SQL statement, the zone may configure up to 20 Columns that contain the formatting definition for displaying the output data.

These parameters are applicable to the zone types

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**). The parameters follow the description pattern of **Column x**.
- Info Data Explorer - Multiple SQLs (**F1-DE**). The parameters follow the description pattern of **Column x for SQL y**.
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**). The parameters follow the description pattern of **Column x for SQL y**.

The following sections describe the various types of mnemonics.

Contents

- [Source Mnemonics](#)
- [Formatting Mnemonics](#)
- [Click Mnemonics](#)

Source Mnemonics

This table describe the mnemonics that control how the data in a column is derived.

Mnemonic	Description	Valid Values	Comments
source=	Defines how the column's value is derived.	SQLCOL	Indicates that the source of the column's value comes from a column in the SQL statement. This type of column must also reference the sqlcol= mnemonic.
		BO	Indicates that the source of the column's value comes from a business object . This type of column must also reference the bo= , input= and output= mnemonics to define how to interact with the business object.
		BS	Indicates that the source of the column's value comes from a business service . This type of column must

Mnemonic	Description	Valid Values	Comments
			also reference the bs= , input= and output= mnemonics to define how to interact with the business service.
SS			Indicates that the source of the column's value comes from a service script . This type of column must also reference the ss= , input= and output= mnemonics to define how to interact with the service script.
FORMULA			Indicates that the source of this column's value is calculated using a formula. This type of column must also reference the formula= mnemonic.
SETFUNC			Indicates that the source of this column's value is calculated using a superset of values from the rows in the SQL statement. This type of column must also reference the setfunc= mnemonic.
ICON			Indicates that the source of this column's value is a display icon reference (meaning that an icon will be displayed in the column). This type of column must also reference the icon= mnemonic to define the icon reference. NOTE: When using this source mnemonic, the formatting mnemonic type= is not applicable.
FKREF			Indicates that the source of this column's value is an FK reference (meaning that the FK reference's context menu and information string will be displayed in the column). This type of column must also reference the fkref= and input= mnemonics to define how the FK reference is called. NOTE: When using this source mnemonic, the formatting mnemonic type= is not applicable.
SPECIFIED			Indicates that the source of this column's value is specified by concatenating literals and other column values. This type of column must also reference the spec= mnemonic.

Mnemonic	Description	Valid Values	Comments
		MSG	Indicates that the source of this column is a message from the message table (along with any substitution variables). This type of column must also reference the msg= mnemonic.
sqlcol=	Defines the column in the SQL statement when source=SQLCOL .	COLUMN_NAME	Enter the name of a column that is retrieved in the SELECT statement. Note that if the select statement uses an alias for a column, then the alias should be referenced here.
		x	Where x is an integer value that references a column by its relative position in the SELECT statement. For example, sqlcol=3 would display the 3rd column in the SELECT statement).
bo=	Defines the business object to invoke when source=BO . This mnemonic must be used in conjunction with the input= and output= mnemonics to define how information is sent to / received from the business object.	Cx	This means business object code is defined in an earlier column. For example, define C1 if column 1 defines the business object.
		COLUMN_NAME	This means the business object was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Business Object Code'	This means the business object is defined directly. For example 'F1-BundleImport'.
bs=	Defines the business service to invoke when source=BS . This mnemonic must be used in conjunction with the input= and output= mnemonics to define how information is sent to/received from the business service.	Cx	This means business service code is defined in an earlier column. For example, define C1 if column 1 defines the business service.
		COLUMN_NAME	This means the business service was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Business Service Code'	> This means the business service is defined directly. For example 'F1-GetCountryStates'.
ss=	Defines the service script to invoke when source=SS . This mnemonic must be used in conjunction with the input= and output= mnemonics to define how information is sent to / received from the service script.	Cx	This means service script code is defined in an earlier column. For example, define C1 if column 1 defines the service script.
		COLUMN_NAME	This means the service script was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.

Mnemonic	Description	Valid Values	Comments
		'Service Script Code'	This means the service script is defined directly. For example 'F1-ReturnBtn'.
fkref=	<p>Defines the FK reference used to retrieve the column's information when source=FKREF.</p> <p>This mnemonic must be used in conjunction with the input= mnemonic to define how information is sent to the FK reference to build the information.</p>	Cx	This means FK reference code is defined in an earlier column. For example, define C1 if column 1 defines the FK reference value.
		COLUMN_NAME	This means the FK reference was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'FK Reference Code'	This means the FK Reference is defined directly. For example 'F1-ROLE'.
formula=	<p>Defines the formula to use when source=FORMULA.</p> <p>Examples:</p> <ul style="list-style-type: none"> formula=C1*.90/C2 formula=(C1/C2)*100 	The formula can contain numeric constants, operators and column references.	<p>For column references, use the format Cx where x represents the column number.</p> <p>Refer to Expression Parser for information about the functions supported.</p>
setfunc=	<p>Defines the function to apply to the rows of a given column when source=SETFUNC.</p>	function(Cx)	<p>Where Cx represents a column whose rows should have the function applied and the function is one of the following</p> <ul style="list-style-type: none"> MAX. This derives the maximum value of all rows in the column. MIN. This derives the minimum value of all rows in the column. TOT. This derives the sum (total value) of all rows in the column. ACC. This derives the cumulative total of all rows up to an including the current row.
input=	<p>This is used to define one or more input fields and values passed to business objects, business services, service scripts, and FK references.</p> <p>The syntax is as follows: [ELEMENT_NAME=ELEMENT_REF ELEMENT_NAME=ELEMENT_REF ...]</p> <p>In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the ELEMENT_NAME, which is the name of the element / field in the target. ELEMENT_REF is the value passed</p>	Cx	Where Cx represents the value of a previous column. If the value to pass is in the first column, reference C1.
		COLUMN_NAME	This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'literal value'	This means a literal value within the single quotes should be passed in.
		userTimeZone	This means the current user's time zone should be passed in. This is typically used with the business service F1-ShiftDateTime to convert data in the storage time zone to the user's time zone for display.

Mnemonic	Description	Valid Values	Comments
	in. The next column indicates the possible values for ELEMENT_REF.	installationTimeZone	This means the installation time zone should be passed in. This is typically used with the business service F1-ShiftDateTime to convert data in the storage time zone to the installation time zone for display.
		Examples:	
		<ul style="list-style-type: none"> • input=[USER_ID=C1] • input=[USER_ID=USER_ID] • input=[input/targetTimeZone=userTimeZone] 	
output=	This is used to define the name of the element retrieved from the business object, business service or service script used to populate this column.	elementName	Example: output=personInfo
pagingkey=	This mnemonic is only applicable when the Enable Paging parameter has been configured. It indicates that this column is one of the keys used by the SQL statement to orchestrate paging through results. This mnemonic can only be specified when the source=SQLCOL . FASTPATH: Refer to Pagination Configuration for more information.	Y N	This is the default, meaning that you don't need to indicate pagingkey=N at all to indicate that the column is not one of the paging keys.

Formatting Mnemonics

This table describe the mnemonics that control how a column is formatted.

Mnemonic	Description	Valid Values	Comments
type=	Defines how the column's value is formatted. NOTE: Icon and Foreign Key columns. The source=source mnemonic may be used to indicate a column should be derived from an icon reference or a foreign key (FK) reference. If you use either of these sources, the type= mnemonic is not relevant as either an icon or a context menu / info string will appear in the column.	STRING DATE TIME DATE/TIME MONEY	Columns of this type capture a string. This is the default value. Columns of this type capture a date. Columns of this type capture a time (in database format) and will be displayed using the user's display profile . Columns of this type capture a date and time (in database format) and will be displayed using the user's display profile . Columns of this type capture a monetary field. This type of column may also reference the cur= mnemonic. If the cur mnemonic is not specified, the currency code on the installation record is used.

Mnemonic	Description	Valid Values	Comments
		NUMBER	Columns of this type capture a numeric field. This type of column may also reference the dec= mnemonic.
label=	Defines the column's override label. The label appears in the column's heading and in the zone's drag and drop area.	FIELD_NAME	Enter a valid field name whose label should be used for the column label. This should always be the option used if multiple languages are needed.
	If this mnemonic is not defined, the system uses the column's default label. The source of a column's default label differs depending on the column's source . Note that some sources don't have a default value and omitting this mnemonic will result in a blank label.	'text'	Defines the text directly.
cur=	Defines the currency code applied when type=MONEY if the installation record's currency should not be used.	Cx	This means currency code value is defined in an earlier column. For example, define C1 if column 1 defines the currency code.
		COLUMN_NAME	This means the currency code was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Currency Code'	This means the currency code is defined directly. For example 'USD'.
dec=	Defines the number of decimal places when type=NUMBER . It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.	nR	Where n is the number of decimal places to show. Suffixing the number of decimal places with R means that the system should round up / down. Simply specifying n (without an R) means that decimal places should be truncated. For example, entering dec=4 will display 4 decimal places and truncate the remainder. NOTE: Formatting only. This mnemonic is only used for formatting, it does not impact the precision used for subsequent calculations. For example, if a column retrieved from the database contains 6 significant digits and dec=0 , the column will be shown with no decimal places (truncated), however any references to the column in subsequent will use 6 decimal places. For example, if the column is referenced in a formula or set function, all 6 decimal places will be used.

Mnemonic	Description	Valid Values	Comments
char=	This mnemonic applies special character(s) to the column's value.	'x[]x'	<p>Where x references the literal value to display and [] defines the relative position of the characters (before or after the value).</p> <p>You need only include the [] if you want to position characters in front of the value. For example, char='%' will place a percent sign after the value. If you want to position the word 'minutes' before a value, enter char='minutes []'. If you want to output a value like BUDGET \$123.12 (YTD), enter char='BUDGET [] (YTD)'.</p>
suppress=	<p>This is used to indicate a column should not be displayed.</p> <p>A column would be suppressed if it's only defined for use by subsequent columns, for example, if there is a formula that derives a column using two other columns. In this scenario, the columns referenced in the formula can be suppressed.</p>	<p>true</p> <hr/> <p>false</p>	<p>This is the default, meaning that you don't need to indicate suppress=false at all to indicate that the field should be shown.</p>
suppressSearch=	This is used to indicate a column should not be displayed when the zone is invoked in search mode only.	<p>true</p> <hr/> <p>false</p>	<p>This is the default, meaning that you don't need to indicate suppressSearch=false at all to indicate that the field should be shown.</p>
suppressExport=	This is used to indicate a column should not be downloaded to Excel.	<p>true</p> <hr/> <p>false</p>	<p>This is the default, meaning that you don't need to indicate suppressExport=false at all to indicate that the field should be included in a download.</p>
width=	This is used to override the width of a column (number of pixels). The default value is the maximum width of any cell in the column.	n	<p>Where n is a number between 0 and 999.</p> <p>NOTE:</p> <p>If there is no available breaking point in the data, the column will be longer than the specified number of pixels.</p> <p>The length of the column's label (which appears in the column's heading) may also make the width wider than specified.</p>
color=	This is used to override the column's text color.	A valid HTML "named" color	<p>For example color=red or color=yellow.</p>

Mnemonic	Description	Valid Values	Comments
		A valid RGB color model combination	For example color=#FF0000 or color=#CCCCCC . Note that the # is required.
bgcolor=	This is used to override the column's background color.	A valid HTML "named" color A valid RGB color model combination	Similar to the color= mnemonic. Similar to the color= mnemonic.
order=	Defines the column's default sort order.	ASC DESC	Indicates that the order is ascending. This is the default meaning that it is not necessary to indicate order=ASC . Indicates that the order is descending.

Click Mnemonics

This table describe the mnemonics that define whether a column value may be clicked and if so, what should happen.

Mnemonic	Description	Valid Values	Comments
navopt=	Defines the navigation option that references the target transaction or script when the user clicks a column. Note, this mnemonic should be used in conjunction with the context= mnemonic to define what information is sent to the navigation option's target transaction. This mnemonic is ignored if source=FKREF because the FK reference code defines the hyperlink's destination.	Cx COLUMN_NAME 'Navigation Option Code'	This means navigation option code is defined in an earlier column. For example, define C1 if column 1 defines the navigation option. This means the navigation option was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause. Example: navopt=MAIN_PORTAL This means the navigation option code is defined directly. For example navopt='userMaint' .
context=	This is used to define one or more context fields and values passed to the target navigation option to go along with the navopt= mnemonic. The syntax is as follows: [FIELD_NAME=FIELD_REF FIELD_NAME=FIELD_REF ...] In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the FIELD_NAME, which is the name of the context field in the navigation option. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.	Cx COLUMN_NAME 'literal value'	Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference C1 . This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause. This means a literal value within the single quotes should be passed in.
bpa=	Indicates that a BPA script should be executed with the user clicks the	Cx	Indicates that the BPA script is defined in a previous column.

Mnemonic	Description	Valid Values	Comments
	<p>column and indicates the BPA to execute.</p> <p>Note, this mnemonic should be used in conjunction with the tempstorage= mnemonic to define the temporary storage values that will be initiated when the script is executed.</p> <p>This mnemonic is ignored if source=FKREF because the FK reference code defines the hyperlink's destination.</p>	<p>COLUMN_NAME</p> <hr/> <p>'BPA Script Code'</p>	<p>This means the BPA script to execute was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.</p> <hr/> <p>This means that the BPA script to execute is defined directly.</p>
tempstorage=	<p>This is used to define how temporary storage variables are initiated when the bpa= mnemonic is used.</p> <p>The syntax is as follows: [FIELD_NAME=FIELD_REF FIELD_NAME=FIELD_REF ...]</p> <p>In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the FIELD_NAME, which is the name of the field in temporary storage. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.</p>	<p>Cx</p> <hr/> <p>COLUMN_NAME</p> <hr/> <p>'literal value'</p>	<p>Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference C1.</p> <hr/> <p>This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.</p> <hr/> <p>This means a literal value within the single quotes should be passed in.</p>
list=	<p>This is used to enable work list capability for this column.</p> <p>You may optionally populate the listdesc= mnemonic to override the text that will be placed in the worklist zone.</p>	true	<p>Setting list=true will cause the work list icon to appear in the column's header. If a user clicks the column, it will populate all the rows in the output into the work list zone.</p> <p>NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (F1-DE), the output may be showing a union of the results of multiple SQL statements. In this case, if some of the SQL statements configure a given column with list=true, but not all, only the data in the cells for the statements that configure this mnemonic are put into the work list when the user clicks the icon.</p>
listdesc=	This is an optional mnemonic when using the list= mnemonic. It can be used to override the text that is placed in the work list zone.	Cx	Where Cx represents the value of a previous column. For example, if the text to use is in the first column, reference C1 .
listbroadcast=	Indicates that the broadcast information for the column is also to	true	Use this setting to turn on the feature.

Mnemonic	Description	Valid Values	Comments
	be made available in the work list zone. This means that the work list can be used to broadcast information to a portal in the same manner as a data explorer.		

Zone Action

Most zone types provided by the product allow for one or more Zone Actions to be defined to appear in the zone header. An action can appear as a hyperlink, icon or button. The action can also be provided as an HTML string.

NOTE: Zone types also include parameters for actions defined at the zone type level using `IMPLEMENTOR_ACTION_n` (Action n) parameters. These are rarely used by the product zone types. The actions defined here override any actions defined on the zone type (if present). The details below apply to the zone type level actions as well.

A zone action is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
type=	This mnemonic defines the appearance of the action in the zone header.	LINK	Indicates that the action is shown as a textual hyperlink.
		ICON	Indicates that the action is shown as a graphical icon.
		BUTTON	Indicates that the action is shown as an HTML button.
		ASIS	Indicates that the parameter will provide the HTML to be used for the action.
action=	This mnemonic defines the action to take when the link/icon/button is clicked. This is ignored when the <code>type=ASIS</code> .	NAVIGATION	Indicates that the action is navigation to a page.
		SCRIPT	Indicates that the action is to run a BPA script.
navopt=	Defines the navigation option to use when the <code>action=NAVIGATION</code> .	'NAV_OPT_CD'	Enter a reference to a valid navigation option in single quotes.
bpa=	Defines the script to run when the <code>action=SCRIPT</code> .	'SCRIPT_CD'	Enter a reference to a valid BPA script in single quotes.
icon=	Indicates the icon to use when <code>type=ICON</code> .	DISP_ICON_CD	Enter a reference to a valid display icon.
		'path'	Enter an explicit path to the icon, for example 'images/gotoZone.gif'.
asis=	This is required when the <code>type=ASIS</code> . This provides the ability to precisely define the HTML you wish to have included in the header. All valid HTML is permitted including the use of "ora" css classes and JavaScript functions.	['HTML']	
label=	By default, the label or tooltip will come from the navigation option or BPA script description. Use this mnemonic to override that label.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Enter the text directly in single quotes.
context=[target1=source1 target2=source2]	This is used to pass context data when navigating to a	FIELD_NAME	Indicates that the value should be taken from the field with

Mnemonic	Description	Valid Values	Comments
	<p>page or executing a BPA script. The mnemonic supports passing multiple values.</p> <p>In each case the target context field or BPA script variable is defined first followed by an equal sign, followed by source data defined using one of the valid values defined in the next column.</p> <p>One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.</p>	<p>xpath</p> <p>'constant'</p>	<p>this name from portal context, global context or the page data model. The mnemonic sourceLoc is used for defining the source.</p> <p>Indicates that the value should be taken from a schema field, represented by the Xpath, displayed in this zone. This is valid when the zone is displaying a UI Map.</p> <p>Indicates that the value defined in single quotes should be passed.</p>
sourceLoc=	<p>This mnemonic defines the source of the FIELD_NAME's value in the context mnemonic.</p> <p>If this mnemonic is left blank, the default behavior is as follows:</p> <ul style="list-style-type: none"> - The portal context is checked. - If no portal context value is found, the global context is checked. - If neither value is available, the field is ignored. 	<p>G</p> <p>P</p> <p>D</p>	<p>Indicates that the field's value is retrieved from the global context.</p> <p>Indicates that the field's value is retrieved from the portal context.</p> <p>Indicates that the field's value is retrieved from the page data model.</p>
class=	<p>Use this mnemonic to override the look and feel of the link / icon / button using a different CSS style.</p>	<p>'className1' 'className2'</p>	<p>Enter one or more classes in single quotes. Multiple class names may be provided.</p>
style=	<p>Use this mnemonic to override the look and feel of the action element using the indicated css style.</p>	<p>Standard style= format.</p>	<p>All allowed css style definitions may be used.</p>

Examples:

- **type=BUTTON action=SCRIPT bpa='F1-SET-USER' context=[USER_ID=USER_ID] label=UPDATE_LBL**
- **type=LINK action=NAVIGATION navopt='gotoUser' context=[USER_ID=path(schema/userdId)]**
- **type=ASIS asis=['Search']**

NOTE: If the zone type has actions defined and there is a desire to simply remove the zone type actions, the Zone Action can be set with the following configuration: **type=ASIS asis=[]**

User Filters

Data explorer zones include the ability to define User filters to allow a user to enter data to restrict the zone's rows and / or columns. The filters may be defined individually using User Filter parameters 1–25. Alternatively, a UI map may be defined for capturing filters. In this case, the map's input fields must be associated with the zone's filters by specifying the **xpath=** mnemonic on the respective User Filter parameters.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

A user filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
name=	This mnemonic is used if the zone's filter should be pre-populated with a value from global context, portal context or broadcast from another zone.	FIELD_NAME	
datasource=	This mnemonic defines the source of the filter's pre-populated value defined in the name mnemonic. If this mnemonic is left blank, the default behavior is as follows: - If the field has been broadcast from another zone, the broadcast value is used. - If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken). - If still no value, the global context is checked. - If still no value, no default value is shown.	G P D	Indicates that the zone should look for the filter value in global context. Indicates that the zone should look for the filter value in portal context. Indicates that the zone should look for the filter value in the page data model.
type=	Defines the visual metaphor used to capture the filter values.	DATE DATE/TIME STRING MONEY NUMBER LOOKUP TABLE CHARTYPE ASIS	Filters of this type capture a date. Filters of this type capture a date and time. Filters of this type capture a string Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic. Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic. Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic. Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic. Filters of this type capture predefined characteristic values for a characteristic type (code and description). This type of filter must also reference the chartype mnemonic. Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.
label=	Defines the filter's label that appears in the zone's description bar and in the input area.	FIELD_NAME 'text'	Enter a valid field name whose label should be used for the filter label. This should always be the option used if multiple languages are needed. Defines the text directly.
cur=	Defines the currency code applied when type=MONEY .	CURRENCY_CD	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when type=NUMBER .	N	It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on

Mnemonic	Description	Valid Values	Comments
			the currency code specified on the installation record.
lookup=	Defines the lookup flag whose values appear when type=LOOKUP .	LOOKUP_FIELD_NAME	Enter a reference to a valid lookup field name.
table=	Defines the admin table whose values appear when type=TABLE .	TABLE_NAME	Enter a reference to a valid control table name.
chartype=	Defines the characteristic type code whose values appear when type=CHARTYPE .	CHAR_TYPE_CD	Enter a reference to a valid characteristic type code.
xpath=	This mnemonic is used in conjunction with a Filter Area UI Map. For each filter, you must specify the xpath to the corresponding UI map schema element.	nameFromUIMap	The type= mnemonic must also be appropriate for the map's input field, otherwise the query's SQL could fail.
likeable=	This mnemonic defines if a likeable search is performed on the entered value when type=STRING .	S	The query will add % to the suffix of the filter value.
		P	The query will add % to the prefix of the filter value.
		PS	The query will add % to the prefix and suffix of the filter value.
divide=	The mnemonic controls if a divider line appears above and/or below the filter. Note, you can specify this parameter twice if you want divider lines placed above and below a filter, e.g., divide=above divide=below .	above	This results in a divider line placed above the filter.
		below	This results in a divider line placed below the filter.
searchField=	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	FIELD_NAME	Enter the field name that exactly matches the searchField name specified in the oraSearchField html element in the UI map.
encrypt=	This mnemonic defines if the user filter is encrypted and needs to be searched by hashed value.	[TBL_NAME,FLD_NAME,WHERE_FLD,WHERE_VALUE] NOTE: The field name referenced here should be the source value of the field. However, the SQL should use the hashed value in its filter.	A valid table name and field name are required. The WHERE_FLD and WHERE_VALUE are optional, but if entered, both are required. Use this to only encrypt the field in FIELD_NAME if another field has a certain value. For example encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,'SSN'] . The WHERE_VALUE may also reference another filter, for example encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,F1] .

Examples:

- **label=F1_NBR_DAYS type=NUMBER**
- **label=F1_SHOW_ALL_REQ_FLG type=LOOKUP lookup=F1_SHOW_ALL_REQ_FLG**
- Filter value where a Filter UI Map is defined and Description is one of the filters. **type=STRING xpath=description likeable=S**
 - **type=STRING label=DESCR likeable=S divide=below**
 - **label=REQ_TYPE_CD type=TABLE table=F1_REQ_TYPE**

Hidden Filters

Data explorer zones include the ability to define Hidden filters to restrict the rows and / or columns that appear in the zone. The following are the potential sources of a hidden filter's value:

- The global area contains the fields whose values are maintained in [global context](#).
- The portal area contains the fields describing the object currently displayed in a portal.

- Other zones on a portal can broadcast information to the portal area, which can then in turn be used by the zone as a hidden filter.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)
- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

A hidden filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
name=	This mnemonic defines the name of the field that needs to be broadcast from other zones or populated in the portal context	FIELD_NAME	
datasource=	This mnemonic defines the source of the hidden filter's value. If this mnemonic is left blank, the default behavior is as follows: - If the field has been broadcast from another zone, the broadcast value is used. - If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken). - If still no value, the global context is checked. - If still no value, the zone appears as per the poprule mnemonic.	G P D	Indicates that the zone should look for the filter value in global context. Indicates that the zone should look for the filter value in portal context. Indicates that the zone should look for the filter value in the page data model.
poprule=	This mnemonic controls what happens if the hidden filter is not present.	R O	Indicates that a value for the filter is required. The zone will be set to the "empty state" and the "please broadcast" message will appear in the zone. This is the default value. Indicates that the value is optional. If no value is required, the zone is still built without that value.
type=	Defines the visual metaphor used to capture the filter values.	DATE DATE/TIME STRING MONEY NUMBER LOOKUP TABLE CHARTYPE	Filters of this type capture a date. Filters of this type capture a date and time. Filters of this type capture a string Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic. Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic. Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic. Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic. Filters of this type capture predefined characteristic values for a characteristic type (code

Mnemonic	Description	Valid Values	Comments
			and description). This type of filter must also reference the chartype mnemonic.
		ASIS	Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.
label=	Defines the filter's label that appears in the zone's description bar.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Defines the text directly.
cur=	Defines the currency code applied when type=MONEY .	CURRENCY_CD	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when type=NUMBER .	N	It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.
lookup=	Defines the lookup flag whose values appear when type=LOOKUP .	LOOKUP_FIELD_NAME	Enter a reference to a valid lookup field name.
table=	Defines the admin table whose values appear when type=TABLE .	TABLE_NAME	Enter a reference to a valid admin table name.
chartype=	Defines the characteristic type code whose values appear when type=CHARTYPE .	CHAR_TYPE_CD	Enter a reference to a valid characteristic type code.
searchField=	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	FIELD_NAME	Enter the field name that exactly matches the searchField name specified in the oraSearchField html element in the UI map.

Multi-Select Action

This parameter defines an action to be included in the action area for multi-selection processing. Note that a multi-selection action can only be used if the Multi Select parameter has been set to YES, which causes a checkbox to appear on each row displayed. The action defined here will trigger against all rows selected by the user via the checkbox.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)
- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

A multi select action has the following mnemonics:

Mnemonic	Description	Valid Values	Comments
script=	This mnemonic defines the script to be invoked when the action is clicked. This is required.	SCR_CD	Enter a reference to a valid BPA script or Service Script.
type=	This mnemonic defines how the action should be rendered.	BUTTON	The action is rendered as a button. This is the default.
		LINK	The action is rendered as hypertext.
		ICON	The action is rendered as a graphic icon. For this option, the icon mnemonic is required.

Mnemonic	Description	Valid Values	Comments
icon=	This mnemonic defines the icon to display when type=ICON .	DISPLAY_ICON_CD	Enter a reference to a valid display icon code.
refresh=	This mnemonic indicates how and if a refresh should occur after the script completes.	NO	Indicates that no refresh is performed. This is the default.
		ZONE	Indicates that a refresh of the zone is performed.
		PORTAL	Indicates that a refresh of the entire portal is performed.
label=	By default, the button label, link text or icon tooltip will come from the script description. Use this mnemonic to override that label.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Enter the text directly in single quotes.
list=	When executing the script, the framework builds an XML list containing information from each row selected. This list must be defined in the script's schema and referenced in this mnemonic.	listElementName	Enter a valid list element name from the script schema.
context=[elementName1=rowData1 elementName2=rowData2]	<p>This mnemonic is used to populate the list with the appropriate information from each selected row. The mnemonic supports passing multiple values.</p> <p>In each case the element in the schema list is defined first followed by an equal sign, followed by information about the data used to populate the element defined using one of the valid values defined in the next column.</p> <p>One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.</p> <p>Example of a schema:</p> <pre><schema> <accountInfo type="list"> <accountId/ > <name /> <amount /> <process /></pre>	Cx	Indicates that the element should be populated with a value in the referenced column parameter.
		Px	Indicates that the element should be populated with a value in the referenced post processing parameter.
		COLUMN_NAME	Indicates that the element should be populated with a value from a column in the SQL statement.
		'constant'	Indicates that the value defined in single quotes should be passed.

Mnemonic	Description	Valid Values	Comments
	<pre></ accountInfo> </schema></pre> <p>Example of list and context mnemonics.</p> <p>list=accountInfo</p> <p>context=[accountId=ACCT_ ID name=C2 amount=P3 process='O']</p>		
class=	Use this mnemonic to override the look and feel of the action using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes to be appended to the standard class(es). Multiple class names may be provided.
style=	Use this mnemonic to override the look and feel of the action element using the indicated css style.	Standard style= format.	All allowed css style definitions may be used.

Pagination Configuration

The various data explorer zones in the product support the ability to configure pagination so that a user can ‘page through’ a large set of results using “previous” and “next” buttons or links.

There are several zone parameters that are impacted when attempting to configure this functionality. The following steps highlight the configuration.

- The **Enable Pagination** parameter must be configured to define the basic setup for pagination for the zone. This parameter defines whether the “previous” and “next” actions are defined as buttons, links or icons and indicates the location of the actions. It also allows an indication as to whether the additional rows are simply appended, rather than shown in a new “page”. Refer to the parameter’s inline help for information about the specific syntax.
- It is recommended that the zone is configured with record count and page information by properly configuring the **Record Count Display** parameter. Refer to the parameter’s inline help for information about the specific syntax.
- Configure the **Number of Rows to Retrieve for SQL** parameter to define the number of records displayed per page. If this parameter is not specified the value in the **Number of Rows to Display** parameter is used.
- Configure the key that will be used for paging so that the system can keep track of the ‘page break’. The data must be sorted by the paging key; as a result, the decision for identifying the paging key must take into account the design for the zone and the data being displayed. In addition, the paging key must be unique to ensure that the page breaks occur correctly. See below for configuration examples.
 - The **SQL Statement** must include additional clauses **PAGENEXT** and **PAGEPREV** based on the paging key. In addition, as mentioned above, the paging key must be used in the ORDER BY clause.
 - The **SQL Column** parameters must define the paging key mnemonic to be used in conjunction with the SQL statement paging clauses.

The following zone types support this capability:

- Info Data Explorer - Single SQL (**F1–DE–SINGLE**).
- Info Data Explorer - Multiple SQLs (**F1–DE**). Note that zones of this type support a union of the results of all the SQL statements. As a result, pagination may only be enabled for zones of this type if a single SQL is used. The system is not able to keep track of the pagination across disparate SQL statements.
- Query Data Explorer - Multiple SQLs (**F1–DE–QUERY**).

- Multi Query Data Explorer (**F1-DE-MULQRY**). Zones of this type do not include configuration for SQL statements or column display. However, they do include configuration for the **Enable Pagination**. This parameter must be configured in order for pagination on the individual zones to work.

NOTE: Zones used for a Business Service. Note that pagination is ignored when invoking a data explorer zone via a business service. In this scenario, the zone will return the first “chunk” of rows as defined by the Number of Rows parameters.

Examples

Simple Paging Key

In this example, the Extendable Lookup Value is defined as Column 1 (C1) and is marked as the paging key. This field is unique for the table and works well as a simple paging key.

```
SELECT A.F1_EXT_LOOKUP_VALUE, A.BUS_OBJ_CD
FROM
  F1_EXT_LOOKUP_VAL A,
  F1_EXT_LOOKUP_VAL_L B
WHERE
A.BUS_OBJ_CD = :H1
AND A.BUS_OBJ_CD = B.BUS_OBJ_CD
AND A.F1_EXT_LOOKUP_VALUE = B.F1_EXT_LOOKUP_VALUE
AND B.LANGUAGE_CD = :LANGUAGE
[(F1) AND UPPER(A.F1_EXT_LOOKUP_VALUE) like UPPER(:F1)]
[(F2) AND ((UPPER(B.DESCR_OVRD) like UPPER(:F2))
OR (B.DESCR_OVRD = ' ' AND UPPER(B.DESCR) like UPPER(:F2)))]
[(PAGENEXT) AND A.F1_EXT_LOOKUP_VALUE > :C1]
[(PAGEPREV) AND A.F1_EXT_LOOKUP_VALUE < :C1]
ORDER BY A.F1_EXT_LOOKUP_VALUE
```

Complex Paging Key

Most queries however do not sort by a unique value. In this case, the paging key needs to be set based on the sorting of the query and should include a unique field, such as the primary key, as the last paging key. In this example, the query is showing results sorted by To Do Type, Role and User. All fields, including the To Do Entry ID (the primary key) are marked as paging keys.

```
SELECT TD_TYPE_CD, ROLE_ID, ASSIGNED_TO, ASSIGNED_DTTM, TD_PRIORITY_FLG, TD_ENTRY_ID
FROM CI_TD_ENTRY
WHERE
ENTRY_STATUS_FLG IN ('O', 'W')
[(F1) and TD_TYPE_CD = :F1]
[(F2) AND ASSIGNED_TO = :F2]
[(F3) AND ROLE_ID = :F3]
[(PAGENEXT) and ((TD_TYPE_CD>:C1) or (TD_TYPE_CD=:C1 and ROLE_ID>:C2) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and
[(PAGEPREV) and ((TD_TYPE_CD<:C1) or (TD_TYPE_CD=:C1 and ROLE_ID<:C2) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and
ORDER BY TD_TYPE_CD, ROLE_ID, ASSIGNED_TO, TD_ENTRY_ID
```

Use Data Explorer for Derived Data

There are times when a design warrants displaying data in a data explorer zone that is not accessible via SQL. For example, perhaps the data is from another system and it requires a web service call. The [JMS Message Browser](#) is another example.

The product provides functionality in the data explorer that allows you to call a script after the user filters are populated but before the SQL is executed. The script can retrieve the data as appropriate, store the data in table format so that the SQL can retrieve the data from the table.

The following points provide more detail:

- Create a service script that retrieves the data as needed. This script should store the retrieved data in a temporary table.
 - The product provides a table that may be used. It is called F1_GENERIC_GTT (Generic Global Temporary Table). There is a business service — Create Global Temporary Table Records (**F1-InsertGTTRecords**) that the service script may call to insert the records.

- Note that if the data is accessed via a web service call, it may be appropriate to execute the web service in a separate session using the business service F1-ExecuteScriptInNewSession to trap errors that may be issued by the web service call and provide a better error.
- In the data explorer zone use this service script in the zone's pre-processing script parameter. If any user or hidden filters should be passed into the script, the parameter supports mnemonics for this purpose. Refer to the parameter's inline help for the supported syntax.
- The SQL for the data explorer should access the temporary table that was populated by the service script.

Configuring Timeline Zones

This topic highlights information related to configuring a [timeline zone](#). The zone type is **F1-TIMELINE**. A timeline zone contains one or more "lines" where each line shows when significant events have occurred. The output of each line is driven by an algorithm configured on a timeline zone. Each algorithm is responsible for retrieving a single type of data. For example, one algorithm may retrieve bills for an account in a given time period whereas another algorithm may retrieve payments for that account for the same time period.

The algorithms to configure for a timeline zone use the **Zone — Timeline** plug-in spot. Please note the following details about the behavior for algorithms for this plug-in spot.

- The timeline algorithm receives all the global context values currently populated. In addition, it receives a start and end date from the zone, based on the time period chosen by the user, along with the maximum number of events that can be reasonably display for the chosen time period. The algorithm should use this information to retrieve data for a given type of transaction related to one or more of the input context values for the provided time period.
- For each event found, the algorithm returns information about the event along with many options that assist the user in getting more detail about each event or acting on an event.
 - Event date
 - Primary key of the record (key / value pairs)
 - FK Reference. With this information, the timeline zone will display the appropriate info string to display in the zone's info area when clicking on the event. In addition, the FK reference identifies the appropriate navigation option to use when a user clicks the info string hypertext to view the record on its maintenance page.
 - Background Color and Text Color to use for the event. (Optional). The algorithm may be configured to provide one color for all events or it may be configured to return different colors for different events based on other factors such as status or priority.
 - Icon use for the event. (Optional). The algorithm may be configured to provide an icon to display adjacent to the event.
 - [BPA script](#) to launch when a user clicks on an event. (Optional). The algorithm may return one or more BPA scripts that a user may launch to act on an event. For example, for an event that has a status of **Error**, perhaps a BPA is provided to walk a user through resolving the error.

When a script is initiated from a timeline, the system puts the prime key of the event into a field in the page data model. The name of the field is the column name(s) of the event's prime key. For example, when a script associated with a payment event is kicked off, the system populates a field called PAY_ID with the prime-key of the selected payment.

Note that your specific edge application may supply algorithm types for a timeline zone as part of the base product. Click [here](#) to see the algorithm types available for this plug-in spot. Although algorithm types may be provided, typically the product does not deliver algorithms because the parameters for the algorithms are driven by a particular implementation's business rules and preferences. As a result, the product will also not deliver pre-configured timeline zones. Please refer to your edge application's documentation for more information about what timeline algorithm types are delivered, if any and recommendations for configuration.

Defining Context-Sensitive Zones

A context-sensitive zone allows you to associate a zone with a specific user-interface transaction. A context-sensitive zone appears at the top of the Dashboard when a user accesses a page for which the zone is specified as the context. For example, when viewing a business object, additional zones are visible that are specific to the business object page.

CAUTION: Make sure that the zone is appropriate for the transaction on which you are specifying it. For example, if your zone requires a business object ID as one of its keys, it should not be displayed on the To Do entry transaction.

Select **Admin > Context Sensitive Zone** to maintain context-sensitive zones.

Description of Page

The **Navigation Key** is a unique identifier of a tab page within the system. **Owner** indicates if this navigation key is owned by the base product or by your implementation (**Customer Modification**).

CAUTION: Important! When introducing a new context sensitive zone, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

The grid contains the list of context-sensitive zones and the sequence in which they appear in the dashboard for the selected navigation key. The grid contains the following fields:

- **Zone** is the name of the zone to display in the Dashboard.
- **Sequence** is the sequence in which the zone is displayed (if multiple context-sensitive zones are defined).
- **Owner** indicates if this context sensitive zone is owned by the base product or by your implementation (**Customer Modification**).

Where Used

A context-sensitive zone displays at the top of the Dashboard whenever a user accesses the transaction for with the zone is specified.

Defining Portals

This transaction is used to define / change portals. An implementation may define their own portals. In addition, an implementation may override some of the settings for base product provided portals.

Portal - Main

Navigate to this page using **Admin > System > Portal**.

Description of Page

Enter a meaningful and unique **Portal** code and **Description**. Please be aware that for [stand-alone portals](#), the Description is the portal's title (i.e., the end-users will see this title whenever they open the portal).

CAUTION: Important! When introducing a new portal, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Owner indicates if this portal is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a portal. This information is display-only.

Type flag indicates whether the portal is a **Standalone Portal**, a **Tab Page Portal** or the **Dashboard**. Refer to [There Are Three Types of Portals](#) for more information.

The following fields are only enabled for **Standalone Portals**:

- **Navigation Option** defines the navigation option that is used to navigate to this portal from menus, scripts and your favorite links. The navigation option is automatically created when a **Standalone Portal** is added.
- You'll find an **Add To Menu** button adjacent. This field is only enabled if the navigation option is not referenced on a menu. When you click this button, a pop-up appears where you define a menu. If you subsequently press **OK**, a menu item is added to the selected menu. This menu item references the portal's navigation option. You can reposition the menu item on the menu by navigating to the [Menu page](#). Refer to [Putting Portals on Menus](#) for more information.
- **Application Services** defines the service used to secure this portal. The application service is automatically created when a **Standalone Portal** is added. Please note that only users with access to this application service will be able to view this portal and its zones. Refer to [Granting Access to A Portal](#) for more information.
- **Show on Portal Preferences** indicates if a user is allowed to have individual control of the zones on this portal. The portal will not appear in the accordion on the user's Portal Preferences page if this value is set to No. Note that an implementation may change this value for a product delivered portal.

The grid contains a list of zones that are available in the portal. Click + to add a new zone to the portal. Click - to remove a zone from the portal. The grid displays the following fields:

- **Zone** is the name of the zone as defined on the Zone page.
- **Description** is a description of the zone as defined on the Zone page.
- **Display** controls whether or not the zone is visible in the portal. For portals that are configured to Show on Portal Preferences, users may override this value for their view of the portal.
- **Initially Collapsed** controls whether or not the zone is initially collapsed in the portal. For portals that are configured to Show on Portal Preferences, users may override this value for their view of the portal.
- **Default Sequence** is the default sequence number for the zone within the portal. It does not need to be unique within the portal. Note that a sequence of zero will appear last, not first, in the portal. For portals that are configured to Show on Portal Preferences, users may override this value for their view of the portal.
- **Override Sequence** can be used by an implementation team to override the Default Sequence value that is set in the base product.
- **Refresh Seconds** defines in seconds how often the zone is automatically refreshed. The minimum valid value is 15. The maximum valid value is 3600 (1 hour). A value of 0 indicates no automatic refresh. Implementers can change this value as needed.
- **Owner** indicates if this portal / zone relationship is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

NOTE: Removing zones from a portal. You cannot remove a base product zone from a base product portal. An implementation may override the Display setting to prevent a zone from displaying on the portal. In addition, you cannot remove a zone if a user has enabled it on their Portal Preferences. To remove a zone from the portal list, first make sure that no user has it enabled in their portal preferences.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PORTAL](#).

Portal - Options

Use this page to maintain a portal's options. Open this page using **Admin > System > Portal** and then navigate to the **Options** page.

Description of Page

The options grid allows you to configure the options that provide additional information related to the portal.

Select the **Option Type** dropdown to define its **Value**. **Detailed Description** may display additional information on the option type.

Set the **Sequence** to **1** unless the option can have more than one value.

Owner indicates if this is owned by the base product or by your implementation (**Customer Modification**).

NOTE: You can add new option types. Your implementation may want to add additional portal option types. To do that, add your new values to the customizable lookup field **PORTAL_OPT_FLG**.

Defining Display Icons

Icons are used to assist users in identifying different types of objects or instructions. A limited number of control tables allow administrative users to select an icon when they are configuring the system. Select **Admin > System > Display Icon Reference** to maintain the population of icons available for selection.

Description of Page

Each icon requires the following information:

- **Display Icon** is a code that uniquely identifies the icon.
- **Icon Type** defines how big the icon is (in pixels). The permissible values are: **30 x 21**, **21 x 21**, and **20 x 14**. Note that only icons that are **20 x 14** can be used on base product instructions.
- **Description** contains a brief description of the icon.
- **URL** describes where the icon is located. Your icons can be located on the product's web server or on an external web server.
- To add a new icon to the product web server, place it under the **/cm/images** directory under the **DefaultWebApp**. Then, in the **URL** field, specify the relative address of the icon. For example, if the icon's file name is **myIcon.gif**, the **URL** would be **/cm/images/myIcon.gif**.
 - If the icon resides on an external web server, the **URL** must be fully qualified (for example, **http://myWebServer/images/myIcon.gif**).
 - **Owner** indicates if this icon is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DISP_ICON](#).

Defining Navigation Keys

Each location to which a user can navigate (e.g., transactions, tab pages, tab menus, online help links, etc.) is identified by a navigation key. A navigation key is a logical identifier for a URL.

Navigation Key Types

There are three types of navigation keys:

- **System navigation keys** define locations where the target for the navigation is a transaction or portal within the system. The navigation keys define the program component that identifies the page to navigate to.
- **External navigation keys** define a URL that identifies the target location. External URLs can be specified as relative to the product web server or fully qualified. External navigation keys always launch in a new instance of a browser window. Examples of external navigation keys include application viewer links and URLs to external systems.

- **Help navigation keys** define an online help topic that identifies the specific page within the online help to launch. Help navigation keys may be related to a program component when the help is related to a specific page in the system.

Navigation Key vs. Navigation Option

The system has two entities that work in conjunction with each other to specify how navigation works:

- **Navigation Key** defines a unique location to which a user can navigate. For example, each page in the system has a unique navigation key. Navigation keys can also define locations that are "outside" of the system. For example, you can create a navigation key that references an external URL. Think of a navigation key as defining "where to go".
- **Navigation Option** defines how a page is opened when a user wants to navigate someplace. For example, you might have a navigation key that identifies a specific page. This navigation key can then be referenced on two navigation options; the first navigation option may allow users to navigate to the page with no context included, while the second navigates to the page with context data provided to automatically display information related to that context.
- Please note that a wide variety of options can be defined on a navigation option. In addition to defining if data is passed to the page, it could also define search options. In addition, there are some navigation options that do not reference a navigation key but rather refer to a BPA script that should be launched.

The Flexibility of Navigation Keys

Navigation keys provide a great deal of functionality to your users. Use navigation keys to:

- Allow users to navigate to new pages or search programs
- Allow users to transfer to an external system or web page. After setting up this data, your users may be able to access this external URL from a menu, a context menu, their favorite links, etc. [Refer to Linking to External Locations](#) for more information.

Refer to the Tool Suite Guide for more information on developing program components.

NOTE: Replacing Base-Package Pages or Searches. If your new page or search has been designed to replace a module in the base-package, the navigation key must indicate that it is [overriding an existing navigation key](#).

Linking to External Locations

If you want to include links to external systems or locations from within the system, you need to:

- Define a **navigation key** that specifies the URL of the location. For example, define an external navigation key that as a URL of <http://www.oracle.com/>.
- Define a **navigation option** that specifies from where in the system a user can go to your external location. For example, define a navigation option with a usage of **Favorites** or with a usage of **Menu**. Your navigation option points to the navigation key you defined above.
- Add your navigation option to the appropriate location within the system. For example, have users add the navigation option to their [Favorite Links](#) or add the navigation option as an item on a [menu](#).

Overriding Navigation Keys

Your implementation may choose to design a program component (e.g., a maintenance transaction or search page) to replace a component provided by the system. When doing this, the new navigation key must indicate that it is overriding the system navigation key. As a result, any menu entry or navigation options that reference this overridden navigation key automatically navigates to the custom component.

For example, if you have a custom On-line Batch Submission page and would like users to use this page rather than the one provided by the system, setting up an override navigation key ensures that if a user chooses to navigate to the On-line Batch Submission from Menu or a context menu, the user is brought to the custom On-line Batch Submission page.

To create an override navigation key, you need to:

- Define a [navigation key](#) using an appropriate [naming convention](#).
- If the Navigation Key Type of the navigation key being overridden is **External**, specify a Navigation Key Type of **Override (Other)** and define the appropriate URL Component.
- If the Navigation Key Type of navigation key being overridden is **System**, specify a Navigation Key Type of **Override (System)** and populate the Program Component ID with your custom program component ID.
- Specify the navigation key that you are overriding in the **Overridden Navigation Key** field.

Refer to the Tool Suite Guide for more information about developing your own program components.

Maintaining Navigation Keys

Select **Admin > System > Navigation Key** to maintain navigation keys.

CAUTION: Important! When introducing a new navigation key, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Description of Page

The **Navigation Key** is a unique name of the navigation key for internal use.

Owner indicates if this navigation key is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

Navigation Key Type includes the following possible values:

- **External** indicates that the location is specified in the **URL Component** field.
- **Help** indicates that the navigation key is used to launch online help where the specific help topic is defined in the **URL Component** field.
- **Override (Other)** indicates that the navigation key overrides another navigation key of type **External** or **Help**. For this option, the name of the navigation key being overridden is populated in the **Overridden Navigation Key** field.
- **Override (System)** indicates that the navigation key overrides a system navigation key. For this option, the name of the navigation key being overridden is populated in the **Overridden Navigation Key** field.
- **System** indicates that the navigation key refers to a transaction in the system identified by its program component.

FASTPATH: Refer to [Navigation Key Types](#) for more information about navigation key types.

FASTPATH: Refer to [Overriding Navigation Keys](#) for more information about settings required to override a system navigation key.

Program Component ID is the name of the program component identified by the key (for system navigation keys). The program component ID can also be used to specify the transaction with which an online help link is associated.

Overridden Navigation Key is the name of the navigation key that the current navigation key is overriding (if **Override (Other)** or **Override (System)** is selected for the **Navigation Key Type**). Refer to [Overriding Navigation Keys](#) for more information.

URL Component is the specific URL or portion of the URL for the navigation key (external and help navigation keys only). The URL can be relative to the product web server or fully qualified.

Open Window Options allows you to specify options (e.g., width and height) for opening a browser window for an external navigation key. (External navigation keys always launch in a new browser window.) You can use any valid features available in the `Window.open()` JavaScript method. The string should be formatted the same way that it would be for the features argument (e.g., **height=600,width=800,resizeable=yes,scrollbars=yes,toolbar=no**). Refer to a JavaScript reference book for a complete list of available features.

Application Service is the application service that is used to secure access to transactions associated with **External** navigation keys. If a user has access to the specified application service, the user can navigate to the URL defined on the navigation key. Refer to [The Big Picture of Application Security](#) for more information.

The grid displays menu items that reference the navigation key (actually, it shows menu items that reference navigations options that, in turn, reference the navigation key).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MD_NAV](#).

Defining Navigation Options

Every time a user navigates to a transaction, the system retrieves a navigation option to determine which transaction should open. For example,

- A navigation option is associated with every menu item. When a user selects a menu item, the system retrieves the related navigation option to determine which transaction to open.
- A navigation option is associated with every [favorite link](#). When a user selects a favorite link, the system retrieves the related navigation option to determine which transaction to open.
- A navigation option is associated with every node in the various trees. When a user clicks a node in a tree, the system retrieves the related navigation option to determine which transaction to open.
- Etc.

Many navigation options are shipped with the base product and cannot be modified as these options support core functionality. As part of your implementation, you may add additional navigation options to support your specific business processes.

Navigation options may also be used to launch a BPA script.

The topics in this section describe how to maintain navigation options.

CAUTION: In order to improve response times, navigation options are cached the first time they are used after a web server is started. If you change a navigation option and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. A special button has been provided on the Main tab of the navigation option transaction that performs this function. Please refer to [Caching Overview](#) for information on the various caches.

Navigation Option - Main

Select **Admin > System > Navigation Option** to maintain a navigation option.

Description of Page

Enter a unique **Navigation Option** code and **Description**.

CAUTION: When introducing a new navigation option, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

The **Flush System Login Info** button is used to flush the cached navigation options so you can use any modified navigation options. Refer to [Caching Overview](#) for more information.

Owner indicates if this navigation option is owned by the base product or by your implementation (**Customer Modification**). This field is display-only. The system sets the owner to **Customer Modification** when you add a navigation option.

NOTE: You may not change navigation options that are owned by the base product.

Use **Navigation Option Type** to define if the navigation option navigates to a **Transaction**, launches a BPA **Script** or opens an **Attachment**.

NOTE: The **Attachment** option type is only applicable to navigation options that are used to launch a file attached to a record in the Attachment maintenance object.

For navigation option types of **script**, indicate the **Script** to launch. You can use the **Context Fields** at the bottom of the page if you want to transfer the contents of specific fields to temporary storage variables available to the script. The script engine creates temporary storage variables with names that match the Context Field names.

For navigation option types of **transaction**, define the **Target Transaction** (navigation key) and optionally a specific **Tab Page** (also a navigation key) if a specific tab on the transaction (other than the Main tab) should be opened when navigating to that transaction.

NOTE: Finding transaction navigation keys. When populating the **Target Transaction** and **Tab Page** you are populating an appropriate navigation key. Because the system has a large number of transactions, we recommend using the "%" metaphor when you search for the transaction identifier. For example, if you want to find the currency maintenance transaction, enter "%currency" in the search criteria.

The additional information depends on whether the target transaction is a fixed page or a portal-based page:

- For portal-based pages:
 - **Navigation Mode** is not applicable and should just be set to **Add Mode**.
 - If navigating to a query portal, by default the query portal will open with the default search option defined. If the navigation should open a different search option, define the **Multi-Query Zone** for that query portal and indicate the **Sub-Query Zone** to open by default. Note that for this configuration, it is common to define **Context Fields** to pre-populate search criteria in the target query zone. When using this configuration, be sure that the target query zone's user filters are defined to populate data from context.
- For fixed pages:
 - **Navigation Mode** indicates if the **Target Transaction** should be opened in **Add Mode** or **Change Mode**.
 - **Add Mode** should be used if the option is used to navigate to a transaction ready to add a new object. You can use the **Context Fields** at the bottom of the page if you want to transfer the contents of specific fields to the transaction when it opens.
 - **Change Mode** is only applicable for fixed pages and should be used if the option is used to navigate to a transaction ready to update an object. You have two ways to define the object to be changed:
 - Define the name of the fields that make up the unique identifier of the object in the **Context Fields** (and make sure to turn on **Key Field** for each such field).
 - Define the **Search Transaction** (navigation key) if you want to open a search window to retrieve an object before the target transaction opens. Select the appropriate **Search Type** to define which search method should be used. The options in the drop down correspond with the sections in the search (where **Main** is the first section, **Alternate** is the 2nd section, **Alternate 2** is the 3rd section, etc.). You should execute the search window in order to determine what each section does.

When you select a **Search Type**, define appropriate **Context Fields** so that the system will try to pre-populate the search transaction with these field values when the search first opens. Keep in mind that if a search is populated

with field values the search is automatically triggered and, if only one object is found that matches the search criteria, it is selected and the search window closes.

- **Search Group** is only visible if the **Development Tools** module is **not turned off**. It is used to define the correlation between fields on the search page and the tab page. You can view a tab page's **Search Groups** by viewing the HTML source and scanning for **allFieldPairs**.

The **Go To Tooltip** is used to specify the label associated with the tool tip that appears when hovering over a **Go To** object. Refer to the **Usage** grid below.

The **Usage** grid defines the objects on which this navigation option is used:

- Choose **Favorites** if the navigation option can be used as a **favorite link**.
- Choose **Menus** if the navigation option can be used as a user's **home page** or as a menu or context menu item.
- Choose **Script** if the navigation option can be used in a **script**.
- Choose **Foreign Key** if the navigation option can be used as a **foreign key reference**.
- Choose **Go To** if the navigation option can be used as a "go to" destination ("go to" destinations are used on Go To buttons, tree nodes, and hyperlinks).
- If your product supports marketing campaigns, you can choose **Campaign** if the navigation option can be used as a "post completion" transaction on a campaign. For more information refer to that product's documentation for campaigns.

The **Context Fields** grid contains the names of the fields whose contents will be passed to the **Target Transaction** or **Script** or used to launch an **Attachment**. The system retrieves the values of these fields from the "current" page and transfers them to the target transaction or to the script's temporary storage. Turn on **Key Field** for each context field that makes up the unique identifier when navigating to a transaction in **Change Mode**.

NOTE: For an **Attachment**, the grid should contain the Attachment ID.

NOTE: Navigating from a Menu. The standard followed for many base main **menu** navigation options for fixed transactions is that navigation options launched from the Menu dropdown are configured with no context.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_NAV_OPT](#).

Navigation Option - Tree

This page contains a tree that shows how a navigation option is used. Select **Admin > System > Navigation Option** and navigate to the **Tree** tab to view this page.

Description of Page

The tree shows every menu item, favorite link, and tree node that references the navigation option. This information is provided to make you aware of the ramifications of changing a navigation option.

Database Tools

This section describes a variety of database tools that are supplied with the your product.

Defining Table Options

The topics in this section describe the transaction that allows you to define metadata for the application's tables.

Table - Main

Navigate using **Admin > Database > Table**.

Description of Page

Table Name is the identifier of this table.

Description contains a brief description of the table.

System Table defines if the table includes rows that are owned by the base product.

Enable Referential Integrity defines if the system performs referential integrity validation when rows in this table are deleted.

Data Group ID is used for internal purposes.

Enable Data Dictionary defines if the table is to be included in the [Data Dictionary](#) application viewer.

Table Type defines if the table is an **External Table**, a **View** or a physical **Table**.

Date / Time Data Type defines if the system shows times on this table in **Local Legal Time** or in **Local Standard Time**. Local Legal Time is the time as adjusted for daylight savings / summer time.

Table Classification Type specifies the category of data the table will hold. This is for information purposes only and is not used by any system processing. Valid values are **Admin System Table**, **Admin Non System Table**, **Master Table**, **Transaction Table**, and **Unclassified**.

Table Volume Type specifies the expected amount of data the table will hold. This is for information purposes only and is not used by any system processing. Valid values are **High Volume**, **Low Volume**, **Medium Volume**, and **Unclassified**. The volume of data in a particular table in the system may differ greatly from one implementation to another based on unique business requirements. The values populated for base product tables are set to volumes that are typical but may not be true for a given implementation. The value may be updated to reflect the situation for a given implementation.

Audit Table is the name of the table on which this table's audit logs are stored. Refer to [The Audit Trail File](#) for more information.

Use **Audit Program Type** to define if the audit program is written in **Java** or **Java (Converted)**, meaning it was converted into Java.

NOTE: **Java (Converted)** program types are not applicable to all products.

Audit Program is the name of the program that is executed to store an audit log. Refer to [Turn On Auditing For a Table](#) for more information.

NOTE: **View the source.** If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#).

Upgrade controls what happens to the rows in this table when the system is upgraded to a new release:

- **Keep** means that the rows on this table are not touched during an upgrade
- **Merge** means that the rows on this table are merged with rows owned by the base product
- **Refresh** means that the rows on this table are deleted and refreshed with rows owned by the base product.

Data Conversion Role controls if / how the table is used by the conversion tool:

- **Convert (Retain PK)** means that the table's rows are populated from the conversion schema and the prime key in the conversion schema is used when the rows are converted. A new key is not assigned by the system.
- **Convert (New PK)** means that the table's rows are populated from the conversion schema and the prime key is reassigned by the system during conversion.

- **Not Converted** means that the table's rows are not managed by the conversion tool.
- **View of Production** means that the conversion tool uses a view of the table in production when accessing the rows in the table. This is commonly used for administrative tables.

A **Language Table** is specified when fields containing descriptions are kept in a child table. The child table keeps a separate record for each language for which a description is translated.

A **Characteristic Entity** is populated if this table is used to capture characteristics and indicates the associated characteristic entity lookup value for this table. When defining characteristic types, you indicate the characteristic entities where that characteristic type is applicable / valid.

A **Key Table** is specified when the prime-key is assigned by the system. This table holds the identity of the prime keys allocated to both live and archived rows.

Type of Key specifies how prime key values are generated when records are added to the table:

- **Other** means a foreign-system allocates the table's prime-key.
- **Sequential** means a sequence number is incremented whenever a record is added to the table. The next number in the sequence determines the key value.
- **System-generated** means a program generates a random key for the record when it is added. If the record's table is the child of another table, it may inherit a portion of the random number from its parent's key.
- **User-defined** means the user specifies the key when a record is added.

Inherited Key Prefix Length defines the number of most significant digits used from a parent record's primary key value to be used as the prefix for a child record's key value. This is only specified when the Type of Key is **System-generated** and the high-order values of the table's key is inherited from the parent table.

Java Table Name. This field is used to identify the entity/Java class name of the class that represents the table in the Java code. It should contain a short "camelCased" name to be used as the name of the entity within the system. It must also be a valid Java name, and must be unique across the system. This name is used as follows:

- As the short class name on all classes in the Java hierarchy for the class: the Impl class, the Gen, and the interface.
- In HQL queries, it is used to identify the hibernate entity being selected.

Caching Regime determines if the table's values should be cached when they are accessed by a batch process. The default value is **Not Cached**. You should select **Cached for Batch** if you know the values in the table will not change during the course of a batch job. For example, currency codes will not change during a batch process. Caching a table's values will reduce unnecessary SQL calls and improve performance.

Key Validation determines if and when keys are checked for uniqueness. The default value is **Always Check Uniqueness**. Select **Check Uniqueness Online Only** when the database constructs the keys in the table, such as in log tables. Select **Never Perform Uniqueness Checking** when you know that the database constructs the keys in the table and that users cannot add rows directly to the table, such as in log parameter tables. This will reduce unnecessary SQL calls and improve performance.

Help URL is the link to the user documentation that describes this table.

Help Text contains additional information about the table.

Extract for Translation is only visible in a development environment. It indicates whether or not the table includes strings that are eligible for product translation.

Translation Context is only visible in a development environment. It is used to provide information to a translator about the nature and purpose of rows in the table.

NOTE: Changes to base owned tables. Only the following attributes of tables that are owned by the product are modifiable: Audit Table, Audit Program Type, Audit Program, Caching Regime, Key Validation and Table Volume Type.

The grid contains an entry for every field on the table. Drilling down on the field takes you to the [Table Field](#) tab where you may modify certain attributes. The following fields may also be modified from the grid: **Description**, **Override Label**, **Audit Delete**, **Audit Insert** and **Audit Update**. Refer to the Table Field tab for descriptions of these fields.

Table - Table Field

Navigate using **Admin > Database > Table** and click the **Table Field** tab.

Description of Page

Field Name is the name of the field. It is followed by its **Java Field Name**.

Field Help Text displays the help text listed for this field on the Field page, if populated.

Nullable, **Required** and **Validate** are for internal use.

Turn on **Audit Delete** if an audit record should be stored for this field when a row is deleted. Refer to [How To Enable Auditing](#) for more information.

Turn on **Audit Insert** if an audit record should be stored for this field when a row is added. Refer to [How To Enable Auditing](#) for more information.

Turn on **Audit Update** if an audit record should be stored for this field when it is changed. Refer to [How To Enable Auditing](#) for more information.

The **Allow Customization** is only applicable if the table is an Admin System Table. It indicates fields that an implementation is allowed to change for a base owned record. Changes to the field value of one of these types of fields by an implementation are maintained when upgrading to a new version of the product.

Standard Time Type is only enabled if the Table indicates that the Date/Time Data Type is **Local Standard Time**. Each field that represent date/time should define if it uses **Logical Standard Time**, **Physical Standard Time** or uses a **Referenced Time Zone**.

Sequence is a unique sequence of this field with respect to other fields on the table.

The **Label** column is used to define a special label for this field when it relates to this table if it should be different from the field's label. Note that this only impacts the label on a fixed page user interface. Labels on portal based user interfaces do not use this information.

The **Override Label** is provided if an implementation wants to override the base-product label.

NOTE: If you want the Override Label to be shown in the [data dictionary](#), you must regenerate the data dictionary.

Help Text contains any additional information about the field with respect to its use on this table.

Extract for Translation is only visible in a development environment. For tables marked to extract for translation, each translatable field on the table should indicate **Yes**.

Translation Context is only visible in a development environment. It is used to provide information to a translator about the nature and purpose of the data in this field on this table.

NOTE: Changes to base owned table / fields. Only the following attributes of table / fields that are owned by the product are modifiable: Audit Delete, Audit Insert, Audit Update, Override Label.

Table - Constraints

Select **Admin > Database > Table** and navigate to the **Constraints** tab to view the constraints defined on the table.

Description of Page

The fields on this page are protected as only the product development group may change them.

This page represents a collection of constraints defined for the table. A constraint is a field (or set of fields) that represents the unique identifier of a given record stored in the table or a field (or set of fields) that represents a given record's relationship to another record in the system.

Constraint ID is a unique identifier of the constraint.

Owner indicates if this is owned by the base package or by your implementation (**Customer Modification**)

Constraint Type Flag defines how the constraint is used in the system:

- **Primary Key** represents the field or set of fields that represent the unique identifier of a record stored in a table.
- **Logical Key** represents an alternate unique identifier of a record based on a different set of fields than the Primary key.
- **Foreign Key** represents a field or set of fields that specifies identifying and non-identifying relationships to other tables in the application. A foreign key constraint references the primary key constraint of another table.
- **Conditional Foreign Key** represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.

When **Enable Referential Integrity** is checked, the system validates the integrity of the constraint when a row in the table is modified.

Referring Constraint Owner indicates if this is owned by the base package or by your implementation (**Customer Modification**).

Referring Constraint ID is the **Primary Key** constraint of another table whose records are referenced by records stored in this table.

Referring Constraint Table displays the table on which the Referring Constraint ID is defined. You can use the adjacent go-to button to open the table.

Additional Conditional SQL Text is only specified when the constraint is a **Conditional Foreign Key**. The SQL represents the condition under which the foreign key represents a relationship to the referring constraint table.

NOTE: Additional Conditional SQL Syntax. When specifying additional conditional SQL text, all table names are prefixed with a pound (#) sign.

The Constraint Field grid at the bottom of the page is for maintaining the field or set of fields that make up this constraint.

Field is the name of the table's field that is a component of the constraint.

Sequence The rank of the field as a component of the constraint.

The Referring Constraint Field grid at the bottom of the page displays the field or set of fields that make up the **Primary key** constraint of the referring constraint.

Field is the name of the table's field that is a component of the referring constraint.

Sequence is the rank of the field as a component of the referring constraint.

Table - Referred by Constraints

Select **Admin > Database > Table** and navigate to the **Referred By Constraints** tab to view the constraints defined on other tables that reference the **Primary Key** constraint of this table.

Description of Page

This page is used to display the collection of constraints defined on other tables that reference the table.

Referred By Constraint Id is the unique identifier of the constraint defined on another table.

Referred By Constraint Owner indicates if this constraint is owned by the base package or by your implementation (**Customer Modification**).

Prime Key Constraint Id is the **Primary Key** constraint of the current table.

Prime Key Owner indicates if this prime key is owned by the base package or by your implementation (**Customer Modification**).

Referred By Constraint Table is the table on which Referred By Constraint Id is defined.

When **Enable Referential Integrity** is checked, the system validates the integrity of the constraint when a row in the table is modified.

The grid at the bottom of the page displays the **Field** and **Sequence** for the fields that make up the constraint defined on the other table.

Defining Field Options

The topics in this section describe the transaction that can be used to view information about a field and to change the name of a field on the various pages in the system.

Field - Main

Open this page using **Admin > Database > Field**.

Description of Page

Field Name uniquely identifies this field.

CAUTION: As described in [System Data Naming Convention](#) for most system data tables, the base product follows a specific naming convention. However, this is not true for the Field table. If you introduce new fields, you must prefix the field with **CM**. If you do not do this, there is a possibility that a future release of the application could introduce a new field with the name you allocated.

Owner indicates if this field is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a field.

Base Field indicates that the field inherits some of its definitions from another field.

Data Type indicates if the type of data the field will hold. Valid values are **Character**, **Character Large Object**, **Date**, **DateTime**, **Number**, **Time**, **Varchar2** and **XML Type**. This field is protected if the field refers to a Base Field.

Ext Data Type or Extended Data Type is used to further define the type of data for certain data types. Valid values are **Currency Source**, **Day of Month**, **Duration**, **Money**, **Month of Year**, **Flag** and **Switch**. This field is protected if the field refers to a Base Field.

Precision defines the length of the field. In the case of variable length fields, it is the maximum length possible. For number fields that include decimal values, the precision includes the decimal values. This field is protected if the field refers to a Base Field.

Scale is only applicable for number fields. It indicates the number of decimal places supported by the field. This field is protected if the field refers to a Base Field.

Sign is only applicable for numbers. It indicates if the data may contain positive or negative numbers.

Value List is only visible for products that had at one point included COBOL. It defines the copybook that includes the list of valid values for the field.

Description contains the label of the field. This is the label of the field that appears on the various pages on which the field is displayed. Note, the field's label can be overridden for a specific table by specifying an Override Label on the [table / field](#) information. However, this override is not used in portal based user interfaces. It is only applicable if the field is displayed on fixed page user interfaces.

Java Field Name is the reference to this field used in Java code.

Override Label is used if an implementation would like to override the label that appear on user interfaces in the system.

CAUTION: For fixed pages, if the field's label is overridden for a specific table, that override takes precedence. In this is the case the override on the [table / field](#) page should be used.

Work Field indicates that the field does not represent a database table column.

Help Text is used to provide field level embedded help to this field. If the field is displayed on a user interface that supports display of embedded help, this text may be displayed.

Use **Override Help Text** to override the existing embedded help text for this field.

Extract for Translation is only visible in a development environment. It indicates whether or not the field and its description should be included in an extract of translatable strings when doing a product translation. This flag may be set to “No” for work fields whose label is not visible to a user on any user interface.

Translation Context is only visible in a development environment. It is used to provide information to a translator about the use of the field's label so that an appropriate translation can be provided.

Field - Tables Using Field

Select **Admin > Database > Field** and navigate to the **Tables Using Field** tab to view the tables that contain a field.

Description of Page

The grid on this page contains the **Tables** that reference the **Field**. You can use the adjacent go to button to open the [Table Maintenance](#) transaction.

Defining Maintenance Object Options

A maintenance object defines the configuration of a given “entity” in the system. It includes the definition of the tables that together capture the physical data for the entity. In addition, the maintenance object includes options that define important information related to the maintenance object that may be accessed for logic throughout the system. Several algorithm plug-in spots are also defined on the maintenance object, allowing for business rules that govern all records for this maintenance object.

Many maintenance objects in the system support the use of business objects to further define configuration and business rules for a given record. Refer to [The Big Picture of Business Objects](#) for more information.

Maintenance Object - Main

Select **Admin > Database > Maintenance Object** to view information about a maintenance object.

Description of Page

Most maintenance objects are provided with the base package. An implementation can introduce custom maintenance objects when needed. Most fields may not be changed if owned by the base package.

Enter a unique **Maintenance Object** name and **Description**. **Owner** indicates if this business object is owned by the base package or by your implementation (**Customer Modification**).

IMPORTANT: If you introduce a new maintenance object, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Service Name is the name of the internal service associated with the maintenance object.

Click the **View XML** hyperlink to view the XML document associated with the maintenance object service in the [Service XML Viewer](#).

Click the **View MO** hyperlink to view the definition of the maintenance object in the [Maintenance Object Viewer](#).

The grid displays the following for each table defined under the maintenance object:

- **Table** is the name of a given table maintained as part of the maintenance object.
- **Table Role** defines the table's place in the maintenance object hierarchy. Only one **Primary** table may be specified within a maintenance object, but the maintenance object may contain many **Child** tables.
- **Parent Constraint ID** specifies the [constraint](#) used to link the table to its parent table within the maintenance object table hierarchy.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

Maintenance Object - Options

Use this page to maintain a maintenance object's options. Open this page using **Admin > Database > Maintenance Object** and then navigate to the **Options** tab.

Description of Page

The options grid allows you to configure the maintenance object to support extensible options.

Select the **Option Type** drop-down to define its **Value**. **Detailed Description** may display additional information on the option type.

Set the **Sequence** to **1** unless the option can have more than one value.

Owner indicates if this is owned by the base package or by your implementation (**Customer Modification**).

NOTE: You can add new option types. Your implementation may want to add additional maintenance option types. For example, your implementation may have plug-in driven logic that would benefit from a new type of option. To do that, add your new values to the customizable lookup field **MAINT_OBJ_OPT_FLG**.

Maintenance Object - Algorithms

Use this page to maintain a maintenance object's algorithms. Open this page using **Admin > Database > Maintenance Object** and then navigate to the **Algorithms** tab.

Description of Page

The **Algorithms** grid contains algorithms that control important functions for instances of this maintenance object. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to [Plug-in Scripts](#) for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes each **System Event**.

System Event	Optional / Required	Description
Audit	Optional	Algorithms of this type are called to notify of any changes to the maintenance object's set of tables. These algorithms are invoked just before the commit at the end of a logical transaction. The system keeps track of what records are added or changed in the course of

System Event	Optional / Required	Description
Determine BO	Optional	<p>a transaction and all MO audit algorithms are executed in order of when each record was first added or updated.</p> <p>Click here to see the algorithm types available for this system event.</p> <p>Algorithm of this type is used to determine the Business Object associated with an instance of the maintenance object. It is necessary to plug in such an algorithm on a Maintenance Object to enable the business object rules functionality.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click here to see the algorithm types available for this system event.</p>
ILM Eligibility	Optional	<p>Algorithms of this type are used for maintenance objects that are enabled for Information Lifecycle Management. They are used to review records that have reached the maximum retention days and evaluate if they are ready to be archived.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click here to see the algorithm types available for this system event.</p>
Information	Optional	<p>We use the term "Maintenance Object Information" to describe the basic information that appears throughout the system to describe an instance of the maintenance object. The data that appears in this information description is constructed using this algorithm.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click here to see the algorithm types available for this system event.</p>
Revision Control	Optional	<p>An algorithm of this type is used to enforce revision control rules when an object is added, changed or deleted. The maintenance object service calls the plug-in once before the object is processed and once more after applying all business object rules. This allows revision rules to take place in proper revision timings.</p> <p>Click here to see the algorithm types available for this system event.</p>
Transition	Optional	<p>The system calls algorithms of this type upon each successful state transition of a business object as well as when it is first created. These are typically used to record the transition on the maintenance object's log.</p> <p>Note that most base maintenance objects are already shipped with an automatic logging of state transitions. In this case you may use these algorithms to override the base logging functionality with your own. Refer to State Transitions are Audited for more information.</p>

System Event	Optional / Required	Description
		Click here to see the algorithm types available for this system event.
Transition Error	Optional	<p>The system calls this type of algorithm when a state transition fails and the business object should be saved in its latest successful state. The algorithm is responsible for logging the transition error somewhere, typically on the maintenance object's log.</p> <p>Notice that in this case, the caller does NOT get an error back but rather the call ends successfully and the exception is recorded somewhere, as per the plug-in logic.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click here to see the algorithm types available for this system event.</p>

NOTE: You can inactivate algorithms on Maintenance Objects. Your implementation may want to inactivate one or more algorithms plugged into the base maintenance object. To do that, go to the options grid on Maintenance Object - Options and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

Maintenance Object - Maintenance Object Tree

You can navigate to the **Maintenance Object Tree** to see an overview of the tables and table relationships associated with the maintenance objects.

Description of Page

This page is dedicated to a [tree](#) that shows the maintenance object's tables as well as [business objects](#), if you have defined any. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

Defining Valid Values

The product provides several options for defining valid values for a column on a table:

- Lookup
- Extendable Lookup
- Control Table

The following provides more information about the functionality of each of the options available for defining valid values for a column.

Lookup

The simplest mechanism for defining valid values for a column on a table is via the Lookup table. This is sometimes referred to as a “simple” lookup to distinguish it from an extendable lookup (described below). Using the lookup table, you can define valid values and their descriptions. When choosing a valid value that is defined by a lookup, a dropdown UI metaphor is used.

The following highlights functionality related to lookups:

- Lookups are associated with a [Field](#). The field is defined as a character data type with an extended data type of **Flag**. The field's label serves as the description for the prompt to select the valid value.

- The lookup code is limited to four characters and must be all uppercase. If there is any functionality where a valid value in the application must match valid values in an external system, the lookup table may not be the appropriate choice.
- The lookup table does not support additional attributes to be defined for each value. This option is only appropriate when a simple code and description pair is needed.
- The product may also use Lookups to define valid values for functionality unrelated to a column on a table. For example, an algorithm plug-in spot may define an input parameter that supports one or more valid values. The plug-in spot may define the valid values using a lookup, allowing for a simple way to validate the value supplied when invoking the algorithm and to document the valid values.

FASTPATH: For more information, refer to [Defining Lookup Options](#).

Extendable Lookup

The extendable lookup provides a way of defining valid values for a column with additional capabilities that are not supported using the Lookup table. When choosing a valid value that is defined by an extendable lookup, a dropdown UI metaphor is used.

The following highlights functionality related to extendable lookups:

- Each Extendable Lookups is defined using a business object.
- A field should be defined for the extendable lookup code. The field defines the label for the lookup code and defines the size of the lookup code. The size is determined based on the business use case. In addition, there are standard fields included in all extendable lookups, including a description, detailed description and an override description (so that implementations can override the description of base delivered values).
- The extendable lookup may define additional information for each value if warranted by the business requirement. See [Additional Attributes](#) for technical information about additional attributes.

FASTPATH: For more information, refer to [Defining Extendable Lookups](#).

Control Table

There may be scenarios where a list of valid values warrants a standalone maintenance object, which is considered an administrative or control table object. When choosing a valid value that is defined by a control, either a dropdown UI metaphor or a search metaphor is used, depending on how it has been designed.

The following points highlight some reasons why this option may be chosen:

- The records require a lifecycle such that BO status is warranted.
- The additional attributes are sophisticated enough that they warrant their own column definition rather than relying on using CLOB or flattened characteristic. For example, if a list of information needs to be captured with several attributes in the list and the information in the list needs to be searchable.

In this situation, if a product has provided a control table for this type of functionality, it will be documented fully in the appropriate functional area. If an implementation determines that a custom control table is warranted, all the standard functionality for a maintenance object is required: database tables, maintenance object metadata, appropriate Java maintenance classes, portals, zones, etc. Refer to the Software Development Kit for more information. No further information is provided in this section for this option.

Defining Lookup Options

Lookup fields may be used to define valid values for a column in a table or for other types of values like parameters to an algorithm.

FASTPATH: Refer to [Defining Valid Values](#) for some background information.

The base product provides many different lookup fields and their values as part of the product. The following points highlight some functionality related to base-package lookups.

- Fields that are owned by the product will typically provide base lookup values. Implementations are not permitted to remove base delivered lookup values. Implementations may be able to add custom values to base owned lookups. This is controlled with the Custom switch on lookup.
- When the custom switch is unchecked, it means that there is functionality controlled by the base values and an implementation may not extend or customize this functionality. An example of this type of lookup is the Data Type field on the [Field](#) table. The system supports a distinct list of data types and an implementation may not add additional values.
- When the custom switch is checked, it means that there is base functionality supplied for the base values but that an implementation can extend the functionality by supplying their own values. An example of this type of lookup is the Access Mode on [Application Service](#). The product provides many values for the access mode lookup, representing various actions a user may perform. Implementations may add their own values to this lookup. Documentation should indicate when functionality may be extended and should highlight the lookup value that can be extended.

CAUTION: Important! If you introduce new lookup values, you must prefix the lookup value code with **X** or **Y**. If you do not do this, there is a possibility that a future release of the application could introduce a new lookup value with the name you allocated.

- There may be some scenarios where the product supplies a base field and base lookup field with no base lookup values supplied. This occurs when the product doesn't have any base functionality driven by the lookup values. Typically this type of lookup is for information or categorization purposes. The configuration guide for the functional area associated with the lookup should include a configuration step regarding defining values for this type of lookup.
- The description of base delivered values may be overridden by an implementation.

An implementation may also identify the need for defining a new lookup field with its values.

Lookup - Main

Select **Admin > Database > Lookup** to maintain lookup values.

Description of Page

Field Name is the name of the field whose lookup values are maintained in the grid. If you need to add a new lookup field, you must first add the lookup field here, then navigate to the [Field](#) page to create a field with a data type of **Character** and an extended data type of **Flag**.

Owner indicates if this lookup field is owned by the base package or by your implementation (**Customer Modification**). This information is display-only.

Custom switch is used to indicate whether you are allowed to add valid values for a lookup field whose owner is not **Customer Modification**.

- If this switch is turned on, you may add new values to the grid for system owned lookup fields.
- If this switch is turned off, you may not add, remove or change any of the values for system owned lookup fields, with the exception of the override description.

This field is always protected for system owned lookup fields because you may not change a field from customizable to non-customizable (or vice versa).

Java Field Name indicates the name of the field as it is referenced in Java code.

The grid contains the lookup values for a specific field. The following fields are visible:

Field Value is the unique identifier of the lookup value. If you add a new value, it must begin with an **X** or **Y** (in order to allow future upgrades to differentiate between your implementation-specific values and base-package values).

Description is the name of the lookup value that appears on the various transactions in the system

Java Value Name indicates the unique identifier of the lookup value as it is referenced in Java code.

Status indicates if the value is **Active** or **Inactive**. The system does not allow **Inactive** values to be used (the reason we allow **Inactive** values is to support historical data that references a value that is no longer valid).

Detailed Description is the detailed description for a lookup value, which is provided in certain cases.

Override Description is provided if your implementation wishes to override the description of the value provided by the product.

NOTE: If you wish the override descriptions of your lookup values to appear in the application viewer, you must [regenerate](#) the data dictionary application viewer background process.

Owner indicates if this lookup value is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add lookup values to a field. This information is display-only.

Defining Extendable Lookups

Extendable lookups are a way of defining valid values that are more sophisticated than simple lookups.

FASTPATH: Refer to [Defining Valid Values](#) for some background information.

The base product provides extendable lookups as part of the product. The following points highlight some functionality related to base-package extendable lookups.

- The base product may supply base extendable lookup values. Implementations are not permitted to remove base delivered extendable lookup values. It is also possible that implementations may be able to add custom values to base owned lookups. If an implementation is not permitted to add lookup values to the base extendable lookup, the extendable lookup's business object will include validation to prevent this. There is no equivalent of the Custom switch that is on the [lookup](#) field.
- There may be some scenarios where the product supplies a base extendable lookup with no base lookup values supplied. This occurs when the product doesn't have any base functionality driven by the extendable lookup values. The configuration guide for the functional area associated with the extendable lookup should include a configuration step regarding defining values for this type of extendable lookup.
- The description of base delivered values may be overridden by an implementation.

Open this page using **Admin > General > Extendable Lookup**.

You are brought to the **Extendable Lookup Query** where you need to search for the extendable lookup object (i.e., its business object).

Once you have found the appropriate extendable lookup, select the value and you are brought to a standard [All-in-One](#) portal that lists the existing lookup values for the extendable lookup. The standard actions for an All-in-One portal are available here.

Defining Additional Attributes

The product provides a few different ways to define additional values for an extendable lookup. Some of the methods are only relevant for base delivered lookup values as they may impact whether or not an implementation can update the values.

The following table highlights the options available and some summary information about what the option provides.

Option	Brief Description	Extendable Lookup Value Searchable by this Attribute?	Base Delivered Value Modifiable?
Element mapped to BO_DATA_AREA	The element is mapped to a CLOB field that allows for base delivered values to be modified.	No	Yes
Element mapped to BASE_BO_DATA_AREA	The element is mapped to a CLOB field that does not allow for base delivered values to be modified.	No	No
Flattened characteristic	The element is defined using the flattened characteristic mechanism.	Yes	No

The following points highlight information from the table above:

- The decision of defining an additional attribute using a CLOB mapping or a flattened characteristic will depend on whether the functionality expects that the lookup will be known when the attribute is needed (in which case a CLOB mapping is appropriate) or if the functionality expects to determine the extendable lookup based on the attribute (in which case, a flattened characteristic is appropriate).
- When the base product defines an extendable lookup with additional attributes and intends to provide base extendable lookup values, it needs to determine whether or not implementations may update the additional attribute or not.
 - If no and the value is mapped to a CLOB, it will map the value to the BASE_BO_DATA_AREA column. This means that implementations will receive an owner mismatch error when attempting to change the value. In addition, upgrading to a new release will replace the value with the base value.
 - If yes and the value is mapped to a CLOB, it will map the value to the BO_DATA_AREA column. This means that implementations will be able to change the value for a base owned record. In addition, upgrading to a new release will not make any changes to the value.
 - For values mapped to a characteristic, the product does not support an implementation changing the value of a base delivered record. If the product would like to support an implementation overriding this type of value, the business object will need to be designed with a corresponding “override” element (also a flattened characteristic), similar to how the product supplies an Override Description field to support an implementation overriding the base product delivered description for a base value. This element will not be delivered with any value and will allow an implementation to populate that value.

NOTE: Note that in this situation, the product functionality that uses this value must cater for the override value.

- All of this detail is only relevant for base provided extendable lookup values. If an implementation adds custom values for a base supplied extendable lookup, all the additional attributes may be populated as appropriate.
- If an implementation defines a custom extendable lookup business object and wants to define an additional attribute using a CLOB, it doesn’t matter which CLOB column is used. Both BO_DATA_AREA and BASE_BO_DATA_AREA provide the same functionality for custom business objects.

The Big Picture Of Audit Trails

The topics in this section describe auditing, enabling auditing for fields, and auditing queries that you can use to view audit records.

Captured Information

When auditing is enabled for a field, the following information is recorded when the field is changed, added and/or deleted (depending on the actions that you are auditing for that field):

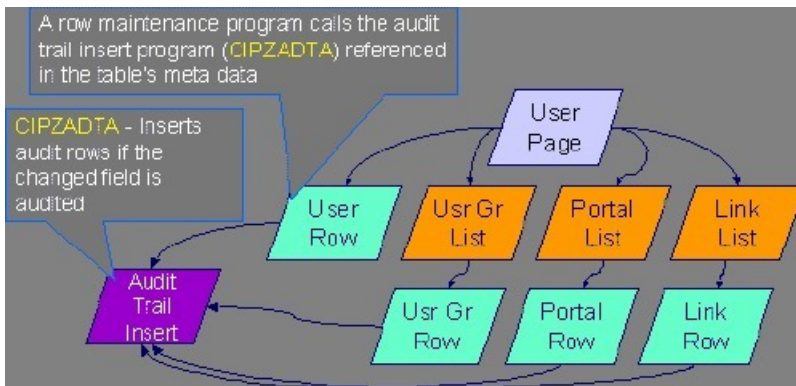
- User ID

- Date and time
- Table name
- Row's prime key value
- Field name
- Before image (blank when a row is added)
- After image (blank when a row is deleted)
- Row action (add, change, delete)

How Auditing Works

You enable auditing on a table in the table's meta-data by specifying the name of the table in which to insert the audit information (the audit table) and the name of the program responsible for inserting the data (the audit trail insert program). Then you define the fields you want to audit by turning on each field's audit switch in the table's field meta-data. You can audit fields for delete, insert and update actions.

Once auditing is enabled for fields in a table, the respective row maintenance program for the table assembles the list of changed fields and calls the audit trail insert program (**CIPZADTA** is supplied with the base package). If any of the changed fields are marked for audit, **CIPZADTA** inserts audit rows into the audit table (**CI_AUDIT** is the audit table supplied with the base package).



NOTE: Customizing Audit Information. You may want to maintain audit information other than what is described in [Captured Information](#) or you may want to maintain it in a different format. For example, you may want to maintain audit information for an entire row instead of a field. If so, your implementation team can use **CIPZADTA** and **CI_AUDIT** as examples when creating your own audit trail insert program and audit table structures.

The Audit Trail File

Audit log records are inserted in the audit tables you define. The base product contains a single such table (called **CI_AUDIT**). However, the audit insert program (**CIPZADTA**) is designed to allow you to use multiple audit tables.

If you want to segregate audit information into multiple tables, you must create these tables. Use the following guidelines when creating new audit tables (that use the **CIPZADTA** audit insert program):

- The new audit tables must look identical to the base table (**CI_AUDIT**).
- The new tables must be prefixed with **CM** (e.g., **CM_AUDIT_A**, **CM_AUDIT_B**, etc.).
- The name of the new table must be referenced on the various tables whose changes should be logged in the new table.

NOTE: Interesting fact. It's important to note if you use your own tables (as opposed to using the base package table called **CI_AUDIT**), the SQL used to insert and access audit trail records (in **CIPZADTA**) is dynamic. Otherwise, if the base package's table is used, the SQL is static.

How To Enable Auditing

Enabling audits is a two-step process:

- First, you must turn on auditing for a table by specifying an audit table and an audit trail insert program.
- Second, you must specify the fields and actions to be audited for the table.

The following topics describe this process.

Turn On Auditing For a Table

In order to tell the system which fields to audit, you must know the name of the table on which the field is located. You must specify the audit table and the audit trail insert program for a table in the table's meta-data.

NOTE: Most of the system's table names are fairly intuitive. For example, the user table is called **SC_USER**, the navigation option table is called **CI_NAV_OPT**, etc. If you cannot find the table using the search facility on the [Table Maintenance](#) page, try using the [Data Dictionary](#). If you still cannot find the name of the table, please contact customer support.

To enable auditing for a table:

- Navigate to the [Table maintenance](#) page and find the table associated with the field(s) for which you want to capture audit information.
- Specify the name of the **Audit Table**.

NOTE: Specifying the Audit Table. You can use the audit table that comes supplied with the base package (**CI_AUDIT**) to audit multiple tables and fields. All the audit logs are combined in a single table (**CI_AUDIT**). However, you can also have a separate audit table for each audited table. Refer to [The Audit Trail File](#) for more information.

- Specify the name of the **Audit Program** (**CIPZADTA** is the default audit program supplied with the base package).

CAUTION: By default, none of a table's fields are marked for audit. Even though you have enabled auditing for a table, you must still specify the fields and actions on those fields to be audited (see below).

Specify The Fields and Actions To Be Audited

The system only audits actions (insert, update and delete) made to fields that you want audited.

To specify the fields and actions to be audited:

- Navigate to the [Table - Table Field maintenance](#) page for a table on which you have enabled auditing.
- For each field you want to audit, specify the actions you want to audit by turning on the **Audit Delete**, **Audit Insert** and **Audit Update** switches as appropriate.

NOTE: Note. You can also turn on the audit switches using the Field grid at the bottom of the [Table maintenance page](#).

CAUTION: Audit Program Caching! The audit program from the table meta-data is read into a program cache on the application server whenever the date changes or when the server starts. If you implement new auditing on a table, your

audit trail does not become effective until this program cache is reloaded. In other words, new audits on tables where the audit program was not previously specified do not become effective until the next day (or the next restart of the application server). However, if you change the fields to be audited for a table where the audit program is already in the cache, your changes are effective immediately.

Audit Queries

There are two queries that can be used to access the audit information.

Audit Query by User

This transaction is used to view changes made by a user that are stored on a given [Audit Trail File](#).

CAUTION: The system only audits changes that you've told it to audit. Refer to [The Big Picture Of Audit Trails](#) for more information.

Navigate to this page by selecting **Admin > Database > Audit Query By User**.

Description of Page

To use this transaction:

- Enter the **User ID** of the user whose changes you wish to view.
- Enter the name of the table on which the audit trail information is stored in **Audit Table**. Refer to [The Audit Trail File](#) for more information about this field.

NOTE: Default Note. If only one audit table is used to store audit trail information, that table is defaulted.

- Specify a date and time range in **Created between** to restrict the records that result from the query.

NOTE: Default Note. The current date is defaulted.

- Click the search button to display all changes recorded on a specific audit table associated with a given user.

Information on this query is initially displayed in reverse chronological order.

The following information is displayed in the grid:

- **Row Creation Date** is the date and time that the change was made.
- **Audited Table Name** contains the name of the table whose contents were changed.
- **Primary Key** is the prime key of the row in the **Audited Table** whose contents were changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Field Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Field Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

Audit Query by Table / Field / Key

This transaction is used to view audited changes made to a given table.

CAUTION: The system only audits changes that you've told it to audit. Refer to [The Big Picture Of Audit Trails](#) for more information.

NOTE: Most of the system's table names are fairly intuitive. For example, the user table is called `SC_USER`, the navigation option table is called `CL_NAV_OPT`, etc. If you cannot find the table using the search facility on the [Table Maintenance](#) page, try using the [Data Dictionary](#). If you still cannot find the name of the table, please contact customer support.

This transaction can be used in several different ways:

- You can view all audited changes to a table. To do this, enter the **Audited Table Name** and leave the other input fields blank.
- You can view all audited changes to a given row in a table (e.g., all changes made to a given user). To do this, enter the **Audited Table Name** and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).
- You can view all audited changes to a given field in a table (e.g., all changes made to all customers' rates). To do this, enter the **Audited Table Name** and the **Audited Field Name**.
- You can view all audited changes to a given field on a specific row. To do this, enter the **Audited Table Name**, the **Audited Field Name**, and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).

Navigate to this page by selecting **Admin > Database > Audit Query By Table/Field/Key**.

Description of Page

To use this transaction:

- Enter the name of the table whose changes you wish to view in **Audited Table Name**.
- If you wish to restrict the audit trail to changes made to a specific field, enter the **Audited Field Name**.
- If you wish to restrict the audit trail to changes made to a given row, enter the row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**). These fields are dynamic based on the **Audited Table Name**.
- Specify a date and time range in **Created between** to restrict the records that result from the query.

NOTE: The current date is defaulted.

- Click the search button to display all changes made to this data.

Information on this query is initially displayed in reverse chronological order by field.

The following information is displayed in the grid:

- **Create Date/Time** is the date / time that the change was made.
- **User Name** is the name of the person who changed the information.
- **Primary Key** is the prime key of the row in the **Audited Table** whose contents where changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

Bundling

The topics in this section describe the bundling features in the application.

About Bundling

Bundling is the process of grouping entities for export or import from one environment to another.

For example, you might export a set of business objects and service scripts from a development environment and import them into a QA environment for testing. The group of entities is referred to as a bundle. You create export bundles in the source environment; you create import bundles in the target environment.

Working with bundles involves the following tasks:

- Configuring entities for bundling if they are not preconfigured
- Creating an export bundle, which contains a list of entities to be exported from the source environment
- Creating an import bundle to import those entities to the target environment
- Applying the import bundle, which adds or updates the bundled entities to the target environment

Sequencing of Objects in a Bundle

Bundle entities are added or updated to the target environment in the sequence defined in the bundle

Typically, the sequence of entities does not matter. However, sequence is important in the following situations:

- Entities that are referenced as foreign keys should be at the top of the sequence, before the entities that reference them. Specify zones last, as they typically contain numerous foreign key references.
- When importing a business object, specify the business object first, then its plug-in scripts, then the algorithms that reference the scripts, and then the algorithm types that reference the algorithms.
- When importing a portal and its zones, specify the portal first and then its zones.
- When importing a multi-query zone, specify the referenced zones first and then the multi-query zone.
- Always specify algorithms types before algorithms.

You can specify the sequence when you define the export bundle or when you import the bundle to the target environment.

Recursive Key References

Recursive foreign keys result when one object has a foreign key reference to another object that in turn has a foreign key reference to the first object.

For example, a zone has foreign keys to its portals, which have foreign keys to their zones. If the objects you want to bundle have recursive relationships, you must create a 'bundling add' business object that has only the minimal number of elements needed to add the entity. A bundling add business object for a zone contains only the zone code and description, with no references to its portals. In the same way, a bundling add business object for a portal defines only its code and description.

When you apply the bundle, the system initially adds the maintenance object based on the elements defined in the bundling add business object. Before committing the bundle, the system updates the maintenance object with the complete set of elements based on its physical business object.

Owner Flags on Bundled Entities

The owner flag of the entities in an import bundle must match the owner flag of the target environment.

If you need to import objects that your source environment does not own, you must replace the owner flag in the import bundle with the owner flag of the target environment.

Configuring Maintenance Objects for Bundling

All base package meta-data objects are pre-configured to support bundling. All other objects must be manually configured.

If a base package maintenance object is pre-configured for bundling, the **Eligible For Bundling** option will be set to "Y" on the Options tab for the maintenance object.

To configure other objects for bundling, review the configuration tasks below and complete all those that apply:

Complete this configuration task...	for...
Make maintenance objects eligible for bundling	All objects to be included in the bundle
Add a foreign key reference	All objects to be included in the bundle
Create a physical business object	All objects to be included in the bundle
Create a bundling add business object	Objects with recursive foreign key references
Add the Current Bundle zone	All objects, if you want the Current Bundle zone to appear on the maintenance object's dashboard. This is not required by the bundling process.
Create a custom Entity Search zone and add it to the Bundle Export portal	All objects, if you want them to be searchable in the Bundle Export portal. This is not required by the bundling process.

Making Maintenance Objects Eligible for Bundling

The "Eligible For Bundling" maintenance object option must be set to "Y" for all bundled objects.

To make maintenance objects eligible for bundling:

1. Go to the [Maintenance Object](#) page and search for the maintenance object.
2. On the Options tab, add a new option with the type **Eligible For Bundling**.
3. Set the value to "Y" and click **Save**.

Adding a Foreign Key Reference

Each maintenance object in a bundle must have a foreign key reference. Bundling zones use the foreign key reference to display the standard information string for the maintenance object.

To add a foreign key reference to the maintenance object:

1. Navigate to [FK Reference](#) and set up a foreign key reference for the primary table of the maintenance object.
2. Navigate to [Maintenance Object](#) and search for the maintenance object.
3. On the **Option** tab, add a new option with the type **Foreign Key Reference**. The value is the name of the foreign key reference you just created.

Creating a Physical Business Object

Each maintenance object in a bundle must have a physical business object. The physical business object's schema represents the complete physical structure of the maintenance object, and includes elements for all fields in the maintenance object's tables. The bundling process uses this schema to generate the XML for the import bundle.

To create a physical business object for the maintenance object:

1. Navigate to [Business Object](#) and specify the maintenance object.

2. Click **Generate** in the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
3. Save the physical business object.
4. Navigate to [Maintenance Object](#) and search for the maintenance object.
5. On the **Option** tab, add a new option with the type **Physical Business Object**. The value is the name of the physical business you just created.

Creating a Bundling Add Business Object

If the objects to be bundled have recursive foreign key references, you must create a bundling add business object to avoid problems with referential integrity.

To create a bundling add business object:

1. Navigate to [Business Object](#) and specify the maintenance object.
2. Click **Generate** in the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
3. Remove all elements that are not essential. Typically, only a code and description are required.
4. Save the physical business object.
5. Navigate to [Maintenance Object](#) and search for the maintenance object you want to bundle.
6. On the **Option** tab, add a new option with the type **Bundling Add BO**. The value is the name of the bundling add business object you just created.

Adding the Current Bundle Zone

If you want the Current Bundle zone to appear on the maintenance object's dashboard, you must add the Current Bundle zone as a context-sensitive zone for the maintenance object.

To add the Current Bundle zone to the maintenance object:

1. Navigate to [Context Sensitive Zone](#) and search for the navigation key for the maintenance object.
2. Add the Current Bundle zone F1-BNDLCTXT, to that navigation key.

Adding a Customized Entity Search Query Zone to the Bundle Export Portal

If you want the maintenance object to be searchable in the Bundle Export portal, you must first create an entity-specific query zone to search for the maintenance object. Then you must create a customized entity search zone that references this new query zone. Finally, you must add the customized entity search zone to the Bundle Export portal.

To make the maintenance object searchable:

1. Create an entity-specific query zone to search for the maintenance object:
 - a) Navigate to [Zone](#) and search for one of the base query zones, such as the Algorithm Search zone F1-BNALGS.
 - b) Click the **Duplicate** button in the page actions toolbar.
 - c) Enter a name for the new zone.
 - d) Click **Save**.
 - e) Locate the **User Filter** parameter in the parameter list. Add SQL to search for the maintenance object(s) you want to appear in the zone.

- f) Save the query zone.
2. Create a customized entity search zone:

This step only needs to be done once. If you already have a customized search zone in the Bundle Export portal go to step 3

 - a) Navigate to [Zone](#) and search for the F1-BNDLENTQ Entity Search zone.
 - b) Duplicate this zone (as described above).
 - c) Remove any references to base query zones.
3. Add the new entity-specific query zone to the customized entity search zone:
 - a) Locate the customized entity search zone for your Bundle Export portal. This is the zone created in Step 2.
 - b) Locate the Query Zone parameter in the parameter list. Add the name of the query zone you created in Step 1.
 - c) Save the entity search zone.
4. Add the customized entity search zone to the Bundle Export portal:

This step needs to be done only once.

 - a) Navigate to [Portal](#) and search for the Bundle Export portal, F1BNDLEM.
 - b) In the zone list, add the entity search zone you created in Step 2. (Add the new zone after the base entity search zone).
 - c) Save the portal.

Working with Bundles

Use the Bundle Export portal to create an export bundle. The export bundle contains a list of entities to be exported from the source environment. When you are ready to import the objects, use the Bundle Import portal to import the objects to the target environment.

Creating Export Bundles

An export bundle contains a list of entities that can be imported into another environment.

To create an export bundle:

1. Log on to the source environment from which objects will be exported.
2. Select **Admin > Implementation Tools > Bundle Export > Add**.
3. Complete the fields in the Main section to define the bundle's basic properties.

NOTE: You can use the Entities section to add bundle entities now, or save the bundle and then add entities as described in step 5.

4. Click **Save** to exit the Edit dialog. The export bundle status is set to Pending.
5. While an export bundle is in Pending state, use any of the following methods to add entities to the bundle:
 - a) Use the **Entity Search** zone on the Bundle Export portal to search for entities and add them to the bundle. If an entity is already in the bundle, you can remove it.
 - b) To import entities from a .CSV file, click **Edit** on the Bundle Export portal, and then click **CSV File to Upload**. Specify the file name and location of the .CSV file containing the list of entities. Click **Submit** to upload the file, and then click **Save** to save the changes.

- c) Use the **Current Bundle** zone in the dashboard of the entity you want to add. (All entities that are configured to support bundling display a Current Bundle zone). This zone displays a list of all pending export bundles to which you can add the entity.
 - d) When you check an entity into revision control, specify the export bundle on the **Revision Info** dialog.
6. When you have added all entities, click **Bundle** in the Bundle Actions zone on the Bundle Export portal. The export bundle state is set to Bundled and the Bundle Details zone displays the XML representation of every entity in the bundle.

NOTE: The owner flags of the entities in the bundle must match the owner flag of the bundle itself. If the owner flags do not match, the system displays a warning message. Click **OK** to continue or **Cancel** to abort the bundle. If you click OK, you will need to resolve the owner flag discrepancy before you import the bundle to the target environment.

7. Copy the XML from the **Bundle Detail** zone to the clipboard (or to a text file). You can now create an import bundle and apply it to the target environment.

NOTE: If you need to make additional changes to the bundle, you must change the bundle state by selecting the **Back to Pending** button in the **Bundle Actions** zone.

Creating and Applying Import Bundles

Import bundles define a group of entities to be added or updated in the target environment.

Before you create an import bundle, you must have already created an export bundle, added entities, and set the bundle's state to Bundled.

To create an import bundle and apply it to the target environment:

1. If you have not already copied the XML from the export bundle, do so now:
 - a) Select **Admin > Implementation Tools > Bundle Export** and search for the bundle.
 - b) Copy the XML from the **Bundle Detail** zone to the clipboard (or to a text file).
2. Log on to the target environment.
3. Select **Admin > Implementation Tools > Bundle Import > Add**.
4. In the **Bundle Actions** zone, click **Edit XML**.
5. Paste the contents of the clipboard (or text file if you created one) into the **Bundle Detail** zone.
6. Make any necessary changes to the XML and click **Save**. The status of the import bundle is set to Pending.

NOTE: Use caution when editing the XML to avoid validation errors.

7. To remove entities from the import bundle or change their sequence, click **Edit**. Enter your changes and click **Save** to exit the Edit dialog.
8. When you are ready to apply the bundle, click **Apply**. The import bundle state is set to Applied and the entities are added or updated in the target environment.

Editing Export Bundles

You can add or remove entities from an export bundle when it is in Pending state. You can also change the sequence of entities.

To edit to an export bundle that has already been bundled, you must change the bundle state by selecting the **Back to Pending** button on the Bundle Export portal.

To edit a pending export bundle:

1. Open the bundle in edit mode.
2. Click **Edit** on the Export Bundle portal.
3. Make any necessary changes on the edit dialog and then click **Save**.

Editing Import Bundles

You can remove entities from an import when it is in Pending state. You can also change the sequence of entities and edit the generated XML.

To edit a pending import bundle:

1. Open the bundle in edit mode.
2. To edit the XML snapshot, click **Edit XML**. Edit the XML code as needed, then click **Save**.

NOTE: Use caution when editing the XML to avoid validation errors.

3. To remove entities or change their sequence, click **Edit**. Make any necessary changes and click **Save**.

Revision Control

The topics in this section describe the revision control features in the application.

About Revision Control

Revision control is a tool provided for the development phase of a project to allow a user to check out an object that is being worked on. In addition, it captures the version of the maintenance object when users check in an update, maintaining a history of the changes to the object.

If revision control is enabled for an object you must check out the object to change it. While the object is checked out no one else can work on it. You can revert all changes made since checking out an object, reinstate an older version of an object, recover a deleted object, and force a check in of an object if someone else has it checked out.

NOTE: Revision control does not keep your work separate from the environment. Because the metadata for maintenance objects is in the central database, any changes you make to an object while it is checked out will be visible to others and may impact their work.

Many of the maintenance objects used as configuration tools are already configured for revision control, but it is turned off by default. For example, business objects, algorithms, data areas, UI maps, and scripts are pre-configured for revision control.

Turning On Revision Control

Revision control is turned off by default for maintenance objects that are configured for revision control.

To turn on revision control:

1. Add the base package **Checked Out** zone to the Dashboard portal.
 - a) Navigate to [Portal](#).
 - b) Search for the portal `CI_DASHBOARD`.
 - c) In the zone list for the **Dashboard** portal, add the zone `F1-USRCHKOUT`.
2. Set up application security.

For users to have access to revision control, they must belong to a user group that has access to the application service `F1-OBJREVBOAS`.
3. Add the revision control algorithm to the maintenance object that you want to have revision control.

This step must be repeated for each maintenance object that you want to have revision control.

 - a) Go to the [Maintenance Object](#) page and search for the maintenance object that you want to have revision control.
 - b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm `F1-REVCTL`.

Configuring Maintenance Objects for Revision Control

Most configuration tool maintenance objects are pre-configured for revision control. You can configure other maintenance objects for revision control, as well.

To configure other objects for revision control:

1. Create a physical business object for the maintenance object.

A physical business object is one that has a schema with elements for all of the fields for the tables in the maintenance object. Follow these steps to create a physical business object:

 - a) Navigate to [Business Object](#) and specify the maintenance object.
 - b) Use the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
 - c) Save the physical business object.
 - d) Go to the [Maintenance Object](#) page and search for the maintenance object for which you want to enable revision control.
 - e) On the **Options** tab of the maintenance object add a new option with the type **Physical Business Object**. The value is the name of the physical business object that you just created.
2. Add a foreign key reference to the maintenance object.

The revision control zones will display the standard information string for the object based on the foreign key reference. Follow these steps to create a foreign key reference:

 - a) Navigate to [FK Reference](#) and set up a foreign key reference for the primary table of the maintenance object.
 - b) Go to the [Maintenance Object](#) page and search for the maintenance object.
 - c) On the **Options** tab of the maintenance object, add a new option with the type **Foreign Key Reference**. The value is the name of the foreign key reference that you just created.
3. Add the **Revision Control** zone to the maintenance object.

- a) Navigate to [Context Sensitive Zone](#) and search for the navigation key for the maintenance object.
 - b) Add the Revision Control zone, F1-OBJREVCTL, to that navigation key.
4. Add the revision control algorithm to the maintenance object.
- a) Go to the [Maintenance Object](#) page and search for the maintenance object that you want to have revision control.
 - b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm F1-REVCTL.

Working with the Revision Control Zones

You use two zones in the dashboard to work with revision controlled objects when revision control is turned on.

The **Revision Control** zone gives you several options for managing the revision of the currently displayed object. This zone also shows when the object was last revised and by whom. This information is linked to the **Revision Control Search** portal which lists all of the versions of the object.

Using the Revision Control zone you can:

- Check out an object in order to change it.
- Check in an object so others will be able to work on it.
- Revert the object back to where it was at the last checkout.
- Force a check in of an object that is checked out by someone else. You need special access rights to force a check in.
- Delete an object.

The **Checked Out** zone lists all of the objects that you currently have checked out. Clicking on an object listed in this zone will take you to the page for that object. The zone is collapsed if you have no objects checked out.

See [Revision Control Search](#) for more information about Check In, Force Check In, and Check Out one or more records simultaneously.

Checking Out an Object

You must check out a revision controlled object in order to change it.

An object must have revision control turned on before you can check it out.

NOTE: When you first create or update an object a dialog box informs you that the object is under revision control. You can select **OK** to check out the object and save your changes, or **Cancel** to stop the update.

1. Go to the object that you want to work on.
2. Select **Check Out** in the **Revision Control** dashboard zone.

Checking In an Object

You must check in a revision controlled object in order to create a new version of it. Checking in an object also allows others to check it out.

1. Select a link in the **Checked Out** dashboard zone to go to the object that you want to check in.
2. Select **Check In** in the **Revision Control** dashboard zone.
3. Provide details about the version:
 - In the **External References** field state the bug number, enhancement number, or a reason for the revision.
 - In the **Detailed Description** field provide additional details regarding the revision.

- In the **Keep Checked Out** box specify if you want to keep the object checked out. If you keep the object checked out then your revision is a new version that you can restore later.
- In the **Add To Bundle** box specify if the object belongs to a bundle.

4. Select **OK** to check in the object.

Reverting Changes

Reverting changes will undo any changes you made since you checked out an object.

To revert changes:

1. Go to the object that you want to revert.
2. Select **Revert** in the **Revision Control** dashboard zone.
3. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Once reverted, the object can be checked out by another user.

Forcing a Check In or Restore

You can force a check in if an object is checked out by another user and that person is not available to check it in.

You must have proper access rights to force a check in or restore.

To force a check in or restore:

1. Go to the object that is checked out by another user.
2. Select **Force Check In** or **Force Restore** in the Revision Control zone.

The **Force Check In** option is the same as a regular check in. The **Force Restore** option checks in the object and restores it to the previously checked in version.

Deleting an Object

If revision control is turned on for an object, you must use the **Revision Control** zone to delete it.

The object must be checked in before it can be deleted.

To delete a revision controlled object:

1. Go to the object that you want to delete.
2. Select **Delete** in the **Revision Control** zone.
3. Provide details regarding the deletion.
4. Select **OK** to delete the object.

The system creates a revision record before the object is deleted so that the deleted object can be restored.

Restoring an Object

You can restore an older version of either a current object or a deleted object.

An object must be checked in before an older version can be restored.

To restore an object:

1. Go to the **Revision History** portal for the object.
If the object was deleted you must search for it by going to **Admin > Implementation Tools > Revision Control**.
2. Select the desired entity by clicking the hyperlink in the **Details** column.
3. Locate the row in the version history that has the version that you want to restore and click **Restore**.
4. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Working with the Revision Control Portal

The **Revision Control** portal lists information about each version of a revision controlled object.

You can navigate to the **Revision Control** portal from either a link in the **Revision Control** dashboard zone or by going to **Revision Control** portal through **Admin**.

If you want to find the Revision History entry for an earlier version or deleted object, you must search for the object using the **Revision Control Search** portal. Once you select the desired entry, you can restore a previous version of the object clicking **Restore** in the row for the version that you want to restore. You can also see the details of each version by clicking the broadcast icon for that version.

See [Working with Revision Control Zones](#) for more information about tasks that can be performed through Revision Control.

Revision Control Search

The **Revision Control Search** portal allows users to search for entities that have a revision history. The **Search By** dropdown provides additional functionality so that users can search for revisions that are associated to theirs or other's user ID. Users can also **Check In**, **Force Check In**, or **Check Out** one or more entities through this portal.

Zone Options

- **Revision History Search** allows the user to query for revised entities based on a combination of criteria.
 - In the **User ID** field, enter the user ID that is associated with a revision.
 - In the **External Reference** field, enter an ID from an external system and is associated with a revision.
 - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
 - In the **Key 1**, **Key 2**, **Key 3**, **Key 4**, **Key 5** fields, enter the primary identifier(s) for the revised entity. Typically, the entity only requires a single key, but some entities require more than one (for example, Oracle Utilities Customer Care and Billing SA Type require CIS Division and SA Type).
 - In the **Status** dropdown menu, select the entity status for your search.
- **Check In** allows the user to search for entities currently checked out to the logged in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
 - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
 - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Force Check In** allows the user to search for entities that are currently checked out by other user IDs (excluding the logged in user ID) based on a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
 - In the **Checked Out By User** field, enter the user ID that has the entity in a Checked Out status.

- In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
- In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Check Out** allows the user to search for entities currently checked in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them out.
- In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
- In the **Key** field, enter the primary identifier(s) for the revised entity.

Please see [Working with Revision Control Zones](#) for more information about working with individual entities.

Information Lifecycle Management

Information Lifecycle Management (ILM) is designed to address data management issues, with a combination of processes and policies so that the appropriate solution can be applied to each phase of the data's lifecycle.

Data lifecycle typically refers to the fact that the most recent data is active in the system. As time progresses, the same data becomes old and unused. Older data becomes overhead to the application not only in terms of storage, but also in terms of performance. This older data's impact can be reduced by using advanced compression techniques, and can be put into slower and cheaper storage media. Depending on how often it's accessed, it can be removed from the system to make an overall savings of cost and performance. The target tables for ILM are transactional tables that have the potential to grow and become voluminous over time.

The Approach to Implementing Information Lifecycle Management

This section describes the product approach to implementing ILM for its maintenance objects (MOs).

NOTE: The term archiving is used to cover any of the possible steps an implementation may take in their data management strategy, including compression, moving to cheaper storage, and removing the data altogether.

Age is the starting point of the ILM product implementation for some of its high volume data. In general "old" records are considered eligible to be archived. In the product solution, maintenance objects (MOs) that are enabled for ILM have an ILM Date on the primary table and the date is typically set to the record's creation date. (An MO may have special business rules for setting this date, in which case, a different date may be used to set the initial ILM Date). For implementations that want to use ILM to manage the records in the MO, the ILM date is used for defining partitions for the primary table.

There are cases where a record's age is not the only factor in determining whether or not it is eligible to be archived. There may be some MOs where an old record is still 'in progress' or 'active' and should not be archived. There may be other MOs where certain records should never be archived. To evaluate archive eligibility using information other than the ILM Date, the ILM enabled MOs include an ILM Archive switch that is used to explicitly mark records that have been evaluated and should be archived. This allows DBAs to monitor partitions based on age and the value of this switch to evaluate data that may be ready to be archived.

Evaluating records to determine their archive eligibility should still occur on "old" records. The expectation is that a large percentage of the old records will be eligible for archiving. The small number that may be ineligible could be updated with a more recent ILM date. This may cause the records to move into a different partition and can delay any further evaluation of those records until more time has passed.

For each MO enabled for ILM, the product provides a batch process to review "old" records and an ILM eligibility algorithm that contains business logic to evaluate the record and mark it eligible for archiving or not. The following sections provide more information about the batch process and algorithm functionality.

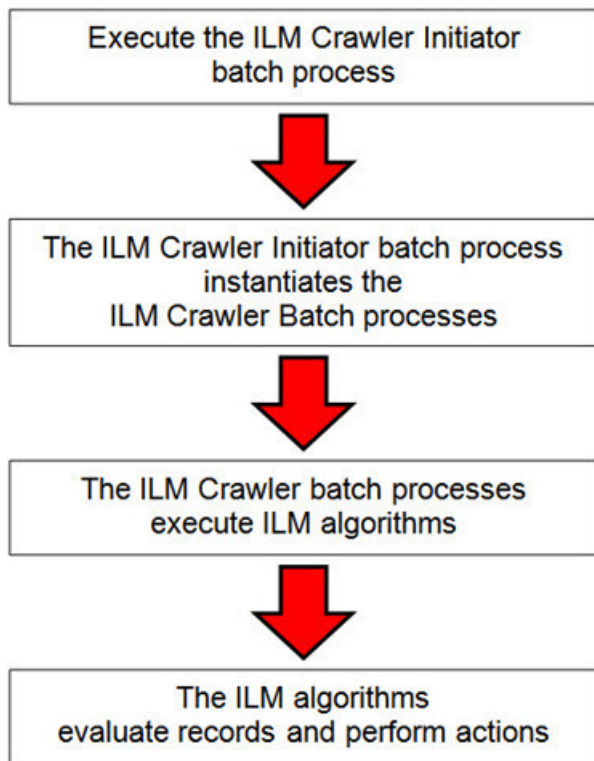
Batch Processes

There are two main types of batch processes that manage data for ILM in the application: ILM Crawler Initiator and individual ILM Crawlers (one for each MO that is configured for ILM).

- **ILM Crawler Initiator: (F1-ILMIN)** - The ILM Crawler Initiator is a *driver* batch process that starts the individual ILM Crawler batch control as defined by the MO's options.

Restartable: In case of server failure, the ILM Crawler Initiator process can be restarted, which will also restart the ILM Crawler processes.

- **ILM Crawler:** Each maintenance object that is configured for ILM defines an ILM Crawler. These are *child* batch processes that can be started either by the ILM Crawler Initiator or by a standalone batch submission.



The ILM Crawler batch process selects records whose retention period has elapsed and invokes the MO's ILM eligibility algorithm to determine if the record is ready to be archived or not. The ILM eligibility algorithm is responsible for setting the record's ILM archive switch to 'Y' and updating the ILM date, if necessary.

The retention period defines the period that records are considered active. It spans the system date and cutoff date (calculated as system date - retention days).

The retention days of an MO is derived as follows:

- If the ILM Retention Days MO option is defined, that is used.
- Otherwise, the Default Retention Days from the ILM Master Configuration record is used.

An error is issued if no retention period is found.

The crawler calculates the cutoff date and selects all records whose ILM archive switch is 'N' and whose ILM date is prior to the cutoff date. Each record returned is subject to ILM eligibility.

If the Override Cutoff Date parameter is supplied, it will be used instead of the calculated cutoff date. An error is issued if the override cutoff date is later than the calculated cutoff date. This parameter is useful if an object has many years of

historic data eligible for archiving. Setting this parameter allows for widening the retention period and therefore limiting the process to a shorter period for initial processing

NOTE: ILM Crawler batch processes are designed not to interfere with current online or batch processing. Because of this, these batch processes can run throughout the day.

NOTE: Before passing the cut-off date to the algorithm, the ILM crawler ensures that the number of days calculated (System Date – override cut-off date) is more than the retention period specified in the MO option or the Master Configuration. If the number of days calculated is **less than** the retention period specified on the MO option or the Master Configuration, then it throws an error.

Eligibility Algorithm

Algorithms are triggered by the ILM batch crawler for the maintenance object. The key responsibility of the ILM algorithm is to determine whether a record can be marked as ready to be archived or not. If a record is determined to be ready for archive, the algorithm should set the ILM Archive switch to Y. If not, the algorithm leaves the switch set to N and may decide to update the ILM Date to something more recent (like the System Date) to ensure that the record does not get evaluated again until the future.

This algorithm is plugged into the [Maintenance Object — Algorithm](#) collection.

Oracle Utilities Application Framework provides the algorithm **ILM Eligibility Based on Status (F1-ILMELIG)** to support the ILM batch crawler. Refer to the algorithm type description for details about how this algorithm works. If a maintenance object has special business rules that are evaluated to determine the eligibility for ILM, a custom algorithm can be created and applied by the implementation team.

Enabling ILM for Supported Maintenance Objects

In order to enable ILM for one or more maintenance objects, several steps are needed in both the configuration and in the database. This section describes some configuration enabled by default and some steps that must be taken in order to fully implement ILM.

There is some configuration enabled by default, but it won't be used unless ILM is fully configured. Each maintenance object that the product has configured for ILM has the following provided out of the box:

- **Special Table Columns:** Maintenance objects that support ILM include two specific columns: ILM Archive Switch (**ILM_ARCH_SW**) and ILM Date (**ILM_DT**).
- **Crawler Batch Process:** A “crawler” batch process is provided for each maintenance object that supports ILM and it is plugged into the MO as an option. Refer to [Batch Processes](#) for more information.
- **ILM Eligibility Algorithm:** Each maintenance object that is configured for ILM defines an [eligibility algorithm](#) that executes the logic to set the ILM Archive switch appropriately. This is plugged in to the MO algorithm collection.

If an implementation decides to implement ILM, there are steps that need to be followed, which are highlighted below.

Create the Master Configuration Record

The first step when enabling ILM is to create the **ILM Configuration**[master configuration](#) record.

The master configuration for ILM Configuration defines global parameters for all ILM eligible maintenance objects. For example, the Default Retention Period. In addition, your product may implement additional configuration. Refer to the inline help for specific details about the information supported for your product's ILM configuration.

In addition, the user interface for this master configuration record displays summary information about all the maintenance objects that are configured to use ILM.

Confirm the Maintenance Objects to Enable

In viewing the list of maintenance objects that support ILM in the ILM master configuration page, your implementation may choose to enable ILM for only a subset of the supported maintenance objects. For example, some of the maintenance objects may not be relevant for your implementation. Or perhaps, the functionality provided by the maintenance object is used, but your implementation does not expect a high volume of data.

For each maintenance object that your implementation has confirmed for ILM, the following steps should be taken:

- Determine if the maintenance object should have a different default retention days than the system wide value defined on the master configuration. If so, use the MO option **ILM Retention Period in Days** to enter an override option for this maintenance object.
- Review the functionality of the ILM Eligibility algorithm provided by the product for this maintenance object. Each algorithm may support additional configuration based on business needs. If your organization has special business rules that aren't satisfied by the algorithm provided by the product, a custom algorithm may be provided to override the base algorithm.

For each maintenance object that your implementation does not want to enable for ILM, inactivate the eligibility algorithm. This will ensure that the ILM Crawler Initiator background process does not submit the crawler batch job for the maintenance object in question.

- Go to the [Maintenance Object — Algorithm](#) tab for each maintenance object and take note of the **ILM Eligibility** algorithm code.
- Go to the [Maintenance Object — Option](#) tab for the same maintenance object and add an option with an option type of **Inactivate Algorithm** and the value set to the ILM eligibility algorithm noted in the previous step.

Database Administrator Tasks

In order to implement ILM for each of the maintenance objects determined above, your database administrator must perform several steps in the database for the tables related to each MO. The following points are a summary of those steps. More detail can be found in the Information Lifecycle Management section of your product's *Database Administration Guide*.

- **Initializing ILM Date:** Your existing tables that are enabled for ILM may not have the ILM Date and ILM Archive switch initialized on all existing records. When choosing to enable ILM, a first step is to initialize this data based on recommendations provided in the DBA guide.
- **Referential Integrity:** The recommended partitioning strategy for child tables in a maintenance object is referential partitioning. In order to implement this, database referential integrity features must be enabled.
- **Partitioning:** This provides a way in which the data can segregate into multiple table partitions and will help in better management of the lifecycle of the data.

Schedule the ILM Crawler Initiator

The final step of enabling the system for ILM is to schedule the ILM crawler initiator **F1-ILMIN** regularly based on your implementation's need. It is recommended to only schedule this batch process once all the required database activities are complete.

Ongoing ILM Tasks

For an environment where ILM is enabled, besides the periodic execution of the ILM crawler batch processes to review and mark records, your database administrator has ongoing tasks.

The DBA reviews and maintains partitions and identifies partitions that may warrant some type of archiving step. The Information Lifecycle Management section of your product's *Database Administration Guide* provides more information for your DBA.

Archived Foreign Keys

If your DBA choose to archive a partition, there may be records in the system that refer to one of the archived records as a foreign key.

When a user attempts to view a record using a hyperlink or drill down mechanism, if the implementation of the navigation uses FK Reference functionality, the system will first check if the record exists. If not, it will display a message to the user indicating that the record has been archived.

Configuration Tools

This section describes tools to facilitate detailed business configuration. The configuration tools allow you to extend both the front-end user interface as well as create and define specialized back-end services.

Business Objects

A [maintenance object](#) defines the physical tables that are used to capture all the possible details for an entity in the system. A business object is tool provided to further define business rules for a maintenance object.

This section provides an overview of business objects and describes how to maintain them.

The Big Picture of Business Objects

The topics in this section describe background topics relevant to business objects.

What Is A Business Object?

A business object (BO) is a powerful tool used throughout the system. For many maintenance objects, a BO is a key attribute of the record used to define the data it captures, its user interface behavior and its business rules. Some business objects support the definition of a lifecycle, capturing different states that a record may go through, allowing for different business rules to be executed along the way. This type of business object is considered the “identifying” or “governing” business object. We will see later that other types of BOs exist that are different from the “identifying” business object.

The use of business objects allows for extensibility and customization of product delivered maintenance objects. There are many options to adjust the behavior of base delivered business objects. In addition, implementations may introduce their own business objects if the base product delivered objects do not meet their business needs.

NOTE: Not all maintenance objects in the product support business objects as a “identifying” or “governing” tool. This is the standard going forward for new maintenance objects. However, there are some maintenance objects created before this became a standard.

A Business Object Has a Schema

A business object has elements. The elements are a logical view of fields and columns in one of the maintenance object’s tables. The structure of a business object is defined using an XML schema. The main purpose of the schema is to identify all the elements from the maintenance object that are included in the business object and map them to the corresponding maintenance object fields. Every element in the BO schema must be stored somewhere in the maintenance object. The BO may not define elements that are derived.

When defining elements for the primary table or the language table (for an administrative object) there is no need to define the name of the physical table in the schema. The system infers this information. The following is a snippet of a schema:

```
<schema>
  <migrationPlan mapField="MIGR_PLAN_CD" suppress="true" isPrimeKey="true"/>
  <bo mapField="BUS_OBJ_CD" fkRef="F1-BUSOB"/>
  <customizationOwner mapField="OWNER_FLG" suppress="input"/>
  <version mapField="VERSION" suppress="true"/>
  <description mapField="DESCR"/>
  <longDescription mapField="DESCRLONG"/>
  ...
```

Many maintenance objects have child table collections (e.g., a collection of names for a person, or a collection of persons on an account). Depending on the requirements, the business object may define the full collection such that the user will maintain the information in a grid. However, the schema also supports “flattening” records in a child table so that they can be treated as if they were singular elements. The following are examples of each:

Example of a child table. This is a snippet of the Instructions collection on the migration plan business object. You can see that the list attribute defines the child table and all elements within it map to the appropriate column in that table.

```
<migrationPlanInstruction type="list" mapChild="F1_MIGR_PLAN_INSTR">
  <migrationPlan mapField="MIGR_PLAN_CD" suppress="true"/>
  <sequence mapField="PLAN_INSTR_SEQ" suppress="true"/>
  <instructionSequence mapField="INSTR_SEQ"/>
  <instructionType mapField="INSTR_TYPE_FLG"/>
  <parentInstructionSequence mapField="PARENT_INSTR_SEQ"/>
  <businessObject mapField="BUS_OBJ_CD" fkRef="F1-BOMO"/>
  ...
```

Example of a simple “flattened” field. The business object for Status Reason includes an element called Usage, which maps to a pre-defined characteristic of type **F1-SRUSG**. The “row” defines which child table is being flattened and the attributes of the row in that child that uniquely identify it.

```
<usage mdField="STATUS_RSN_USAGE" mapField="CHAR_VAL">
  <row mapChild="F1_BUS_OBJ_STATUS_RSN_CHAR">
    <CHAR_TYPE_CD is="F1-SRUSG"/>
    <SEQ_NUM is="1"/>
  </row>
</usage>
```

Example of a “flattened row”. This business object for Account includes a single row for the Person collection where only the “financially responsible, main” customer is defined. The “accountPerson” attribute defines one field from that row (the Person Id) and includes the ‘flattening’ criteria in the “row” information. In addition, a second field from that same row (“accountRelType”) is defined. Instead of having to repeat the flattening criteria, the “rowRef” attribute identifies the element that includes the flattening.

```
<accountPerson mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true"/>
    <FIN_RESP_SW default="true"/>
  </row>
</accountPerson>
<accountRelType mapField="ACCT_REL_TYPE_CD" rowRef="accountPerson" dataType="string"/>
```

Example of a “flattened list”. The business object for Tax Bill Type includes an list of valid algorithms for “bill completion”. The Sequence and the Algorithm are presented in a list. The list element identifies the child table and the ‘rowFilter’ identifies the information about the list that is common.

```
<taxBillCompletion type="list" mapChild="C1_TAX_BILL_TYPE_ALG">
  <rowFilter suppress="true" private="true">
    <TAX_BILL_TYPE_SEVT_FLG is="C1BC"/>
  </rowFilter>
  <sequence mapField="SEQ_NUM"/>
  <algorithm mapField="ALG_CD" fkRef="F1-ALG"/>
</taxBillCompletion>
```

In addition, many maintenance objects support an XML structure field within the entity. These fields may be of data type CLOB or XML. One or more business object elements may be mapped to the MO's XML structure field. These elements may be defined in different logical places in the business object schema based on what makes sense for the business rules.

When updating the MO, the system builds a type of XML document that includes all the elements mapped to the XML structure and stores it in one column. The following is an example of elements mapped to an XML column:

```
<filePath mdField="F1_FILE_PATH" mapXML="MST_CONFIG_DATA" required="true"/>
<characterEncoding mdField="F1_CHAR_ENCODING" mapXML="MST_CONFIG_DATA"/>
```

NOTE: If the MO's XML structure field is of the data type XML, the database will allow searching for records based on that data, assuming appropriate indexes are defined. If the MO's XML structure field is of the data type CLOB, indexing or joining to elements in this column via an SQL statement is not typically supported. Note that most MOs are currently using the CLOB data type for the XML structure column, if provided.

Some business objects may have child tables that allow data to be stored in an XML structure field. The schema language supports defining elements from those fields in your schema as well.

Besides including information about the physical "mapping" of the element to its appropriate table / field location in the maintenance object, the schema supports additional syntax to provide the ability to define basic validation and data manipulation rules, including:

- Identifying the primary key of the record or the primary key of the a row in a list.
- Identifying which elements are required when adding or changing a record.
- Default values when no data is supplied on an Add.
- For elements that are lookup values, the lookup may be specified to validate that the value of the element is a valid lookup value.
- For elements that are foreign keys to another table, the FK Reference may be specified to validate the data.

The system will check the validity of the data based on the schema definition obviating the need for any special algorithm to check this validation.

In addition, the schema language may include some attributes that are used to auto-render the view of the record on the user interface, such as the **suppress** attribute. Refer to [BO Defines its User Interface](#) for more information.

NOTE: Refer to [Schema Syntax](#) for the complete list of the XML nodes and attributes available to you when you construct a schema.

A business object's schema may include a subset of the fields and tables defined in the maintenance object. There are two reasons for this:

- The fields or tables may not be applicable to the type of record the business object is governing. For example, a field that is specific to gas may not be included on a Device business object that is specific to electric meters.
- The information is not maintained through the business object, but rather maintained separately. For example, many BO based maintenance objects include a Log table. The records in the log table are typically not included the BO because they are viewed and maintained separately from the business object.

A Business Object May Define Business Rules

A business object may define business rules that govern the behavior of entities of this type.

- Simple element-level validation is supported by schema attributes. Note that element-level validation is executed before any maintenance object processing. For more sophisticated rules you create **Validation** algorithms and associate them with your business object. BO validation algorithms are only executed after "core validation" held in the MO is passed.
- A **Pre-Processing** algorithm may be used to "massage" a business object's elements prior to any maintenance object processing. For example, although simple element-level defaulting is supported by schema attributes, you may use this type of algorithm to default element values that are more sophisticated.
- A **Post-Processing** algorithm may be used to perform additional steps such as creating a To Do Entry or add a log record as part of the business object logical transaction. These plug-ins are executed after all the validation rules are executed.

- An **Audit** algorithm may be used to audit changes made to entities of this type. Any time a business entity is added, changed or deleted, the system detects and summarizes the list of changes that took place in that transaction and hands it over to **Audit** plug-ins associated with the business object. These plug-ins are executed after all the post-processing rules are executed. It is the responsibility of such algorithms to log the changes if and where appropriate, for example as a log entry or an entry in an audit trail table or an entry in the [business event log](#)

By default all elements of the business object are subject to auditing. You can however mark certain elements to be excluded from the auditing process using the **noAudit** schema attribute. Marking an element as not auditable will prevent it from ever appearing as a changed element in the business object's audit plug-in spot. In addition, if the only elements that changed in a BO are ones marked to not audit, the audit algorithm is not even called. Refer to [Schema Syntax](#) for more information on this attribute.

Refer to [Business Object - Algorithms](#) for more information on the various types of algorithms.

The system applies business object rules (schema based and algorithms) whenever a business object instance is added, changed or deleted. This is only possible when the call is made via the maintenance object service. For example, when made via business object interaction ("invoke BO"), the MO's maintenance page or inbound web services that reference the BO. In addition the system must be able to [determine the identifying business object](#) associated with the actual object being processed. If the business object cannot be determined for a maintenance object instance business object rules are not applied.

NOTE:

Pre-Processing is special. The pre-processing algorithm plug-in spot is unique in that it only applies during a BO interaction. It is executed prior to any maintenance object processing. It means that when performing add, change or delete via the maintenance object service, the pre-processing plug-in is not executed.

CAUTION: Direct entity updates bypass business rules! As mentioned above, it is the maintenance object service layer that applies business object rules. Processes that directly update entities not via the maintenance object service bypass any business object rules you may have configured.

FASTPATH: Refer to [BO Algorithm Execution Order](#) for a summary of when these algorithms are executed with respect to lifecycle algorithms.

The plug-in spots described above are available for all business objects and they are executed by the system when processing adds or updates to the business object. It is possible for a specific maintenance object to define a special plug-in spot for business objects of that MO. When this happens, the maintenance object identifies the special algorithm entity lookup value as an [MO option](#): **Valid BO System Event**, causing the BO Algorithm collection to include that system event in its list.

A Business Object Defines its User Interface

One of the responsibilities of an identifying business object is to define its user interface rules for viewing and maintenance of its record. The standard implementation for maintaining a business object is that a [maintenance portal](#) is used to display a record. This portal includes a "map" zone that displays the information about the business object. To add or make changes to a record, the user clicks a button that launches a maintenance BPA script which displays a maintenance "map".

The display and maintenance "maps" are driven by the business object. The BO may define a full [UI map](#) where all the information is displayed based on the map's HTML. Note that for a child BO, the maps may be [inherited](#) by a parent BO (or any BO "up the chain").

The standard going forward is to use schema definition and UI Hints to define user interface behavior so that a full UI map is not needed but rather the HTML is derived. The schema language includes some basic display attributes such as **label** and **suppress**. UI hints provide many additional tags and elements that allow dynamic generation of formatted UI Maps. For more complex behavior in the user interface, for example where javascript is needed, UI map fragments may be defined within the schema via UI hints. In this way only complex UI behavior warrants small snippets of javascript and HTML.

However the rendering of standard fields can be dynamically rendered. UI map fragments also allow for derived fields to be included in the user interface.

A business object schema may include [data areas](#) for segments of its schema definition to allow for reusable components. In this case the data area would also include schema attributes and UI hints for the elements that it is including.

NOTE: Refer to [UI Hint Syntax](#) for detailed information about the supported syntax.

As mentioned in [Business Object Inheritance](#), schemas are not inherited on a child business object. As such, when using UI hints for automatic UI rendering, the child BO must define the full schema with all the definitions. A good business object hierarchy will be designed for reuse meaning that the child BO will include the parent BO schema or alternatively, the BO schemas will include reusable data areas.

Invoking A Business Object

We have talked about defining a business object. This section describes how business objects are used throughout the system to view, add and update records.

- Various parameters for zones that are used to display data in the system include support for retrieving data by referencing a business object. The zone code will “invoke” the BO, meaning that the record will be retrieved using the referenced BO.
- The system’s [scripting](#) language includes a step type to “invoke BO”. This allows for BPA scripts, service scripts and plug-in scripts to retrieve information and add or update records using BO interaction.
- [Inbound web services](#) may reference a business object in its operations collection. This allows external systems to add or update records in our product via web service interaction.

Often when configuring a zone or writing a script, the BO to use in the “invoke BO” statement should be the identifying BO of the record. As such, often the script will include steps prior to the “invoke BO” step to “[determine the identifying BO of the record](#)” and once the identifying BO is found, the script step will invoke that BO. Note that zones and inbound web services reference a BO directly. In each case, if the BO to use should be dynamic, then the zone / inbound web service should reference a service script that can perform the steps to identify the BO and then invoke that BO.

It should be noted however that the BO used in an “invoke BO” statement (or referenced in an inbound web service) **does not have to match** the identifying BO for the record. Here are some examples of where this may be true:

- A script may only require a subset of elements for a record and not the entire record. In this case, it is better for performance purposes to define a special BO (sometimes called a “lite” BO or a “mini” BO) that only defines the needed elements. When the system retrieves the data, it will only access the tables that are included in the BO’s schema definition. In addition, if there are no elements that map to an XML structure field, the system will skip any parsing of that column. Similarly, if a script is **updating** a subset of elements on a record, it may be beneficial to use a “mini” BO to do the updates.

NOTE: Please note the following with respect to using a mini BO. This BO is only used for its schema. This type of BO would not define algorithms or a lifecycle. Because the BO is special, it often should not be able to be used as any record’s identifying BO. To control that, these BOs are often configured to not allow new instances. Refer to [Determine the Identifying BO](#) for more information.

- The maintenance object to be added or updated in a script may not support business objects as “identifying BOs”. For example, Batch Control maintenance object does not have an identifying BO. However, scripts may still wish to retrieve data (or make updates) to these types of records. An easy way to achieve that goal is to define a business object and use “invoke BO” to access the data.

NOTE: Not all maintenance objects support being maintained through a business object interaction. This is true in a small number of older objects where the underlying maintenance service includes additional functionality besides

simply updating the database tables. These maintenance objects are identified via the [MO option](#) **BO Maintenance**, set to **N**.

- Some functionality may be trying to add or update records for a maintenance object in a ‘physical’ manner and do not want or need to use the object’s identifying BO. Or the MO may not have an identifying BO. For example, revision control takes a snapshot of a record for audit purpose and to be able to restore a previous version. In this case, the system wants to capture a full “physical” view of the record. To do this, a special “physical” BO may be created that includes all (or most of) the columns and the child tables.

NOTE: Like the mini BO, the physical BO would not define algorithms or a lifecycle and should not be able to be used as any record’s identifying BO. To control that, these BOs are often configured to not allow new instances. Refer to [Determine the Identifying BO](#) for more information.

NOTE: To reiterate, the BO referenced in the “invoke BO” statement or referenced in an inbound web service does not have to match the identifying BO and does not have to be configured to “allow new instances”.

Determine the Identifying BO

As mentioned in other topics, the identifying BO is the business object that governs the business rules for a record. This is the business object that the record will be validated against when any additions or changes are made to the record as long as updates are made via the maintenance service. This includes using “invoke BO” for add or update, using inbound web service interaction and for access to the maintenance page service (via an old style fixed page or via a business service).

How does the system determine the identifying BO? An algorithm plugged into the maintenance object (the **Determine BO** plug-in spot) is responsible for this. If the maintenance object is not configured with an algorithm for this plug-in spot, or no BO is found by the algorithm, no BO business rules are applied.

Most maintenance objects in the system capture the record’s identifying BO directly on the record. However, it is possible to define the identifying BO somewhere else. For example, there may be some maintenance objects that are master or transaction objects with an associated “type” object where the identifying BO is defined on its “type” object. Note that the standard Determine BO algorithm plugged into most maintenance objects (**F1-STD-DTMBO — Determine Standard Business Object**) checks for these two conditions.

There may also be cases where a single identifying BO is used for all BOs for a given MO. This may be an option used for some older maintenance object created prior to the business object functionality when implementations wish to introduce custom business rules that are common for all records of that MO. The product provides a base algorithm type (**F1-MOBO — Determine Specific Business Object**) that captures the BO as a parameter.

Base Business Objects

For each maintenance object (MO) that supports an “identifying” business object, the type of business object provided by the product depends on the functionality and expected use by implementations. The following are some common patterns.

- There are MOs where the product provides base BOs that implementations may use if applicable for their business rules. In addition, it is expected that implementation will define custom BOs to support their business needs. Good examples of this type of MO are any of the various “rule” MOs. For example, calculation rule in Oracle Utilities Customer Care and Billing or the usage rule in Oracle Utilities Meter Data Management or the form rule in Oracle Public Sector Revenue Management. The product provides business objects for common rules but each implementation could have special rules that they need to implement and will need to create custom business objects.
- There are MOs where the product provides base BOs that supply common behavior for an object. Implementations may find that supplied the business objects match their business requirements and use the BOs as is. It is expected, however that for many implementations, their business rules will require additional elements to be captured or have special rules to apply. In this case the base business objects may be extended. This scenario may apply to ‘master’ data objects in various products such as the Device or Meter or Tax Role.

- There are MOs where the product may deliver a base BO that is not expected to satisfy most implementations because different jurisdictions or different implementations will typically have their own rules. In this case the base delivered BO can be used as a template or starting point for custom defined BOs. Some examples of this are Rebate Claim in Oracle Utilities Customer Care and Billing or the Appeal object in Oracle Public Sector Revenue Management.
- There are MOs where the expectation is that every implementation will have different requirements for the type of data to capture and the product will not supply base BOs that can be used as the “identifying” BO. However, it may supply a “parent” BO that defines the lifecycle and many of the business rules that it expects all records to follow. In these scenarios, the implementations will create “child” BOs that will serve as the “identifying” BOs and refer to the base “parent” BO for many of its rules through [inheritance](#). Some examples of the are Tax Form in Oracle Public Sector Revenue Management or Activity in Oracle Utilities Mobile Workforce Management.
- There are some scenarios where the base product provides business objects and the expectation is that implementations will use the business objects as delivered with little or no customization. This is a case where the system used business objects to implement product functionality, not because there is an expectation that the implementers will extend the functionality, but because the business object model is the favored development tool even for the product. The objects delivered for Configuration Migration Assistant are an example.

NOTE: Not all maintenance objects in the product support business objects as a “identifying” or “governing” tool. This is the standard going forward for new maintenance objects. However, there are some maintenance objects created before this became a standard.

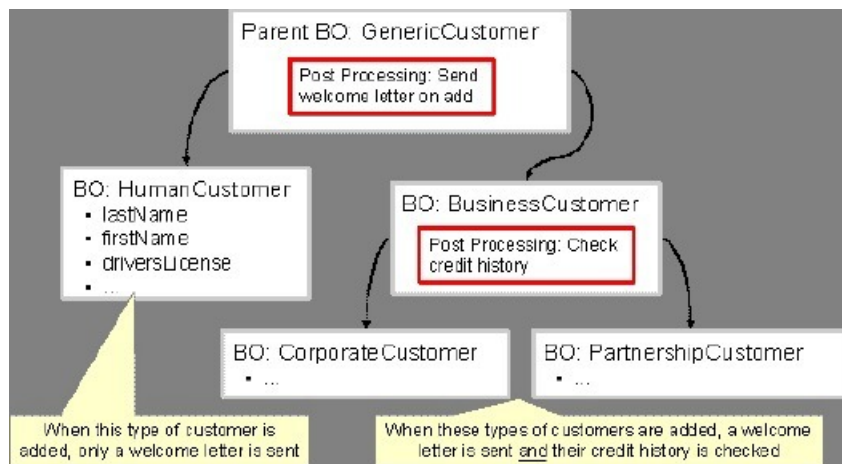
For all maintenance objects, the base product may provide additional BOs that are not meant to be “identifying” BOs, but instead are provided to support functionality to interact with the MO using the BO as a tool as described in [Invoking a BO](#).

- One or more “mini” or “lite” BOs may be supplied for a maintenance object. This may be found when the product has functionality to retrieve a subset of elements for the maintenance object via scripting or via a user interface.
- A “physical” BO may be supplied. This a BO that typically includes all tables and all fields of the maintenance object in there “physical” form. In other words, there is no “flattening” of child tables and any XML structure fields are defined as a single field. Physical BOs are used in system processing where the full record needs to be captured as is. Some functionality that uses a physical BO includes [bundling](#), [revision control](#) and the pre-compare algorithm for CMA to [adjust data prior to comparing](#).

Business Object Inheritance

A business object may inherit business rules from another business object by referencing the latter as its parent. A child business object can also have children, and so on. A parent's rules automatically apply to all of its children (no compilation - it's immediate). A child business object can always introduce rules of its own but never remove or bypass an inherited rule.

The following is an illustration of multiple levels of business object inheritance.



Notice how the "Business Customer" business object extends its parent rules to also enforce a credit history check on all types of customers associated with its child business objects.

Most types of business object system events allows for multiple algorithms to be executed. For example, you can have multiple **Validation** algorithms defined for a business object. For these, the system executes all algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

Other types of system events allows for a single algorithm to be executed. For example, you can only have one **Information** algorithm to format the standard description of a business object instance. For these, the system executes the one at the level nearest to the business object currently being processed.

NOTE: The parent and its children must reference the same maintenance object.

NOTE: Data structures are not inherited. While you can declare schemas on parent business objects, their children will not inherit them. A good practice is to design child business object schemas to **include** their parent business object's schema.

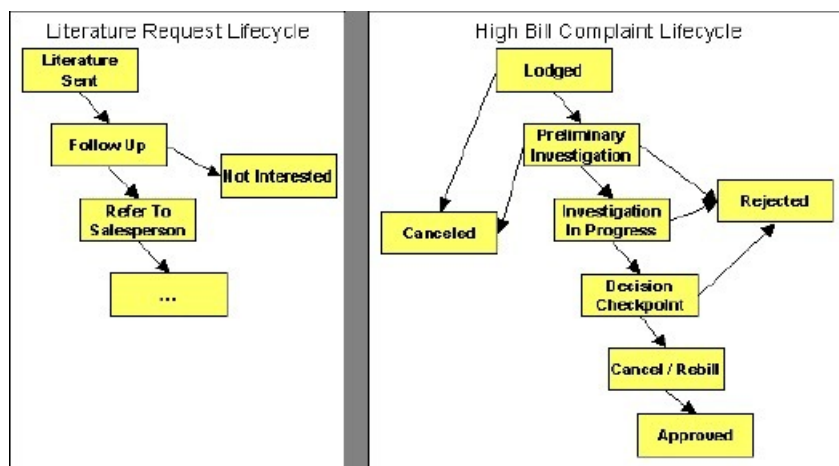
NOTE: User interface maps are inherited. When determining if the business object has a UI map to use for UI rendering, the system looks for display and maintenance maps linked to the BO as options. If the identifying BO does not have maps defined, the system follows "up the chain" of inheritance until it finds a map to use. This allows for a child BO to be used to extend business rules of a parent BO but inherit its user interface behavior. Refer to [Business Object Defines its User Interface](#) for more information.

NOTE: Use Inheritance Wisely. While it is intellectually attractive to abstract behavior into parent BOs to avoid redundant logic and simplify maintenance, before doing this weigh the reuse benefits against the cost in transparency, as it is not easy to maintain a complex hierarchy of business objects.

Each Business Object Can Have A Different Lifecycle

Many maintenance objects have a status column that holds the business entity's current state within its lifecycle. Rules that govern lifecycle state transition (e.g., what is its initial state, when can it transition to another state, etc.) and the behavior associated with each state are referred to as lifecycle rules. Older Maintenance Objects, such as To Do Entry, have predefined lifecycles whose rules are governed by the base-package and cannot be changed. The lifecycle of newer Maintenance Objects exists in business object meta-data and as such considered softly defined. This allows you to have completely different lifecycle rules for business objects belonging to the same maintenance object.

Here are examples of two business objects with different lifecycles that belong to the same maintenance object.



NOTE: A Maintenance Object supports soft lifecycle rules if it is defined with a **Status Field Maintenance Object option**.

The topics that follow describe important lifecycle oriented concepts.

Valid States versus State Transition Rules

The boxes in the above diagram show the potential valid states a business entity of the above business object can be in. The lines between the boxes indicate the state transition rules. These rules govern the states it can move to while in a given state. For example, the above diagram indicates a high bill complaint that's in the **Lodged** state can be either **Canceled** or moved into the **Preliminary Investigation** state.

When you set up a business object, you define both its valid states and the state transition rules.

One Initial State and Multiple Final States

When you set up lifecycle states, you must pick one as the initial state. The initial state is the state assigned to new entities associated with the business object. For example, the above high-bill complaint business object defines an initial state of **Lodged**, whereas the literature request one defines an initial state of **Literature Sent**.

You must also define which statuses are considered to be "final". Typically when an entity reaches a "final" state, its lifecycle is considered complete and no further processing is necessary.

NOTE: Allowing An Entity To Be "Reopened". You can set up your state transition rules to allow a business entity to be "reopened" (i.e., to be moved from a final state to a non-final state). Neither of the above examples allows this, but it is possible if you configure your business object accordingly.

State-Specific Business Rules

For each state in a business object's lifecycle, you can define the following types of business rules.

FASTPATH: Refer to [BO Algorithm Execution Order](#) for a summary of when these lifecycle algorithms are executed with respect to BO level algorithms.

Logic To Take Place When Entering A State

You can define algorithms that execute before a business entity enters a given state. For example, you could develop an algorithm that requires a cancellation reason before an entity is allowed to enter the **Canceled** state.

You can also incorporate state auto-transitioning logic within this type of algorithms. Refer to [auto-transition](#) for more information.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for an enter algorithm to indicate that the other algorithms should be executed by the batch process by setting the "force post processing" indicator to true.

Logic To Take Place When Exiting A State

You can define algorithms that execute when a business entity exists a given state. For example, you could develop an algorithm that clears out error messages when a given entity exits the **Error** state.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for an exit algorithm to indicate that the other algorithms should be executed by the batch process by setting the "force post processing" indicator to true.

Monitor Rules

You can define algorithms to monitor a business entity while it is in a given state. This type of logic is typically used to check if the conditions necessary to [transition](#) the entity to another state exist (and, if so, transition it). For example,

transition an entity to the **Canceled** state if it's been in the **Error** state too long. Another common use is to perform ancillary work while an entity is in a given state. For example, update statistics held on the object while it's in the **Active** state.

Monitor algorithms are invoked when a business entity first enters a state and periodically after that in **batch**. You have the option to defer the monitoring of a specific state until a specific monitoring batch job runs. You do so by associating the state with a specific monitoring process. In this case the system will only execute the monitoring rules of this state when that specific batch process runs. This is useful when processing one type of record typically creates another type of record. You may want the processing of the second set of records to be deferred to a later time.

A monitor algorithm can carry out any business logic. In addition it can optionally tell the system to do either of the following:

- Stop monitoring and transition to another state. The system will not call any further monitoring algorithm plugged in on the state and attempt to transition the entity to the requested new state.
- Stop monitoring. Same as above except that no transition takes place. You may want to use this option to prevent transitions while some condition is true.

If none of the above is requested the system keeps executing subsequent monitoring algorithms.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for a monitor algorithm to indicate that the other algorithms should be executed by the batch process by setting the “force post processing” indicator to true.

FASTPATH: Refer to [Business Object - Lifecycle](#) for more information on how to set up state-specific algorithms.

Inheriting Lifecycle

If a business object references a parent business object, it always **inherits** its lifecycle from the highest-level business object in the hierarchy. In other words, only the highest-level parent business object can define the lifecycle and the valid state transitions for each state. Child business objects, in all levels, may still extend the business rules for a given state by introducing their own state-specific algorithms.

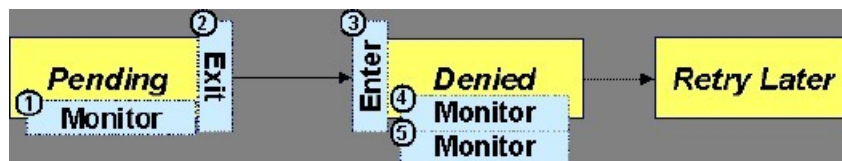
The system executes all state-specific algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

Auto-Transition

In a single transition from one state to another, the system first executes the **Exit** algorithms of the current state, transitions the entity to the new state, executes the **Enter** algorithms of the new state followed by its **Monitor** algorithms. At this point if a **Monitor** algorithm determines that the entity should be further automatically transitioned to another state the remaining monitoring algorithm defined for the current state are NOT executed and the system initiates yet another transition cycle.

Notice that an **Enter** algorithm can also tell the system to automatically transition the entity to another state. In this case the remaining **Enter** algorithm as well as all **Monitor** algorithms defined for the current state are NOT executed.

The following illustration provides an example of an auto-transition chain of events.



In this example a business entity is in a Pending state. While in that state a **Monitor** algorithm determines to auto-transition it to the Denied state. At this point the following takes place:

- No further **Monitor** algorithms of the Pending state are executed
- Pending state **Exit** algorithms are executed

- The system transitions the entity to the Denied state
- Denied state **Enter** algorithms are executed. No further auto-transition is requested.
- Denied state **Monitor** algorithms are executed. No further auto-transition is requested.

Keeping An Entity In Its Last Successful State

By default, any error encountered while transitioning a business entity from one state to another rolls back ALL changes leaving the entity in its original state.

When applicable, the Maintenance Object can be configured to always keep an entity in its last successful state rather than rolling all the way back to the original state. This practice is often referred to as "taking save-points". In case of an error, the entity is rolled back to the last successfully entered state and the error is logged on the [maintenance object's log](#). Notice that with this approach no error is returned to the calling process, the error is just logged.

The logic to properly log the error is in a **Transition ErrorMaintenance Object plug-in**. The system considers a maintenance object to practice "save-points" when such an algorithm is plugged into it.

Even if the maintenance object practices "save-points", in case of an error the system will not keep an entity in the last successfully entered state if that state is either marked as [transitory](#) or one of its **Enter** algorithms has determined that the entity should proceed to a next state. The system will roll back to the first previous state that does not match these conditions.

Monitoring Batch Processes

A monitor batch process may be used to transition a business object into its next state by executing the monitor algorithms associated with the current state of the entity. The use cases for performing the monitor logic in batch are as follows:

- The record may be waiting for something else to occur before transitioning. The monitor algorithm may be coded to determine if the condition is satisfied and initiate the transition then. For example perhaps when entering a state, a field activity is generated and the record should exit the state when the field activity is complete. The monitor algorithm can check the status of the field activity.
- Perhaps a record is added or updated manually and the next step in the BO lifecycle includes a large amount of processing such that the logic should occur in batch. In this case the BO status is configured with an explicit reference to a batch control (called a "deferred" batch control), which indicates to the system that the monitor algorithms should not be performed automatically (but should be deferred to batch). Later when the batch process runs, it selects all the records to process to progress the records.

NOTE: When a status includes a deferred batch control, it may also be configured to allow a user to manually transition the record to the next state rather than waiting for batch. When a user manually transitions a record that includes monitor algorithms, those algorithms are not executed.

- Perhaps a record is added or updated in batch, but a subsequent step in the overall lifecycle should be processed later. This may be accomplished by ensuring that the batch control linked to the state to process later does not match the batch control that added or updated the record.
- Monitor processes may also be used to periodically perform some logic related to the record without actually transitioning the record.

Note that only the parent business object may refer to a deferred monitor batch process. However, any business object in the "[inheritance](#)" chain may be configured with monitor algorithms, which will all be executed.

The base package provides a periodic monitoring batch process for each maintenance object that supports a configurable BO lifecycle. The process periodically executes the monitoring algorithms associated with the current state of an entity, excluding states explicitly referencing a deferred monitoring batch process that is for a different batch control.

A deferred monitoring process works in the same way except that it considers entities whose current state references this particular batch control as their monitor process in addition to records that don't refer to any batch controls as their monitor process. Deferred monitoring is only needed when a given state should not execute its monitor algorithms immediately upon entering the state, but rather when the batch process is specifically executed.

NOTE: MO option configuration. The maintenance object includes options to indicate the batch controls delivered for periodic and deferred monitor batch controls.

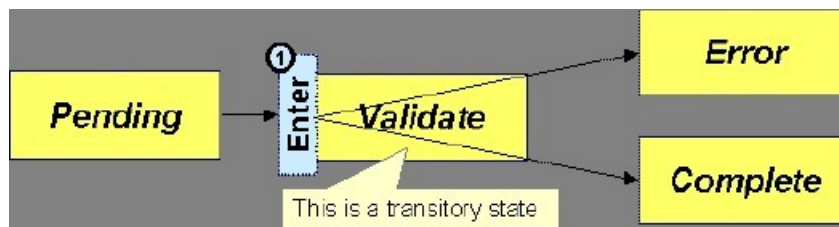
Your business rules will dictate the execution frequency of each monitoring process and the order in which they should be scheduled. Refer to [Monitor Background Processes](#) in the background process chapter for more information about the parameters supported for this type of batch process.

NOTE: Updates to the business object. When the monitor algorithms indicate that the business object should transition, the monitor batch processes are responsible for ensuring the business object is transitioned appropriately and that the appropriate exit, enter and monitor algorithms are executed. Please note that the business object is not updated using a call to the maintenance object service and therefore by default the [business rules](#) plugged in to the business object are not executed. However, it is possible for an Enter algorithm, Exit algorithm or Monitor algorithm to indicate that the other algorithms should be executed by the batch process. If the "force post processing" indicator is set to true, then the batch process invokes the BO Post Processing, BO Audit and MO Audit algorithms.

Transitory States

You can define a state as **Transitory** if you do not wish the business entity to ever exist in that particular state.

The following illustrates a lifecycle with a transitory Validate state.



In this example, the business entity is saved still not validated in the Pending state. At some point, the user is ready to submit the entity for validation and transitions it into a transitory state Validate whose **Enter** rules contain the validation logic. The responsibility of the transitory state's **Enter** algorithms is to decide if the entity is valid or in error and then transitions it into the appropriate final state. In this scenario, you may not ever want the business entity to exist in the Validate state.

Let's also assume that the maintenance object in this example is practicing "[save-points](#)" and requires the entity to be kept in its last successful state. If an error were to occur during the transition from **Validate** to the next state, the system would roll back the entity back to Pending, and not Validate even though the entity has successfully entered the Validate state. Refer to the [Auto Transition](#) section for more information.

State Transitions Are Audited

Most Maintenance Objects that support soft lifecycle definition also have a log to hold significant events throughout a business entity's lifecycle. For example, log entries are created to record:

- When the business entity is created (and who created it)
- When its status changes (and who changed it)
- If a transition error occurred (and the error message)
- References to other objects created throughout the entity's lifecycle. For example, if a To Do entry is created as part of processing the entity, the To Do Entry is referenced in the log.

- Manual entries added by a user (think of these as "diary" entries)

When a business entity is first created and when it transitions into a new state the system calls **Transition** algorithm(s) plugged in on the [Maintenance Object](#) to record these events. If the maintenance object supports a log these events can be captured as log entries.

NOTE: Most base package maintenance objects supporting a log may already provide state transition logging as part of their core logic. In this case you only need to provide a **Transition** plug-in if you wish to override base logging logic with your own.

Required Elements Before Entering A State

You can define additional elements that are required before a business entity can enter a given state. For example, let's assume that a Cancel Reason must be defined before an object can enter the *Canceled* state. You do this by indicating that element as a **Required Element** [state-specific option](#) on the appropriate state on the business object.

Capturing a Reason for Entering a State

Some business objects support configuring certain states to allow or require a status reason when an object enters the state. The product provides a centralized status reason table that may be used to define the valid BO status reasons for various business objects and various states. The status reasons are defined using the [Status Reason](#) portal.

The following sections provide additional information about the BO status reason functionality.

Maintenance Object Must Support Status Reason

In order for a business object to use the centralized status reason table to define reasons, the maintenance object must first support the status reason. MOs that support status reason have the following characteristics:

- The primary table includes a column for Status Reason. This represents the status reason for the record's current status, if applicable.
- The log table includes a column for Status Reason. The standard logic for capturing a log record when entering a state also captures the status reason, if applicable. This allows a user to review the history of the changes in status and the status reason captured for a previous state transition, if applicable.
- The maintenance object option collection includes an option that defines the Status Reason field. This setting is a trigger for business objects of this MO to be able to configure states to allow or require status reason.

Business Object State Indicates if Reasons are Applicable

Once the MO is configured to support status reason, configuration on the business object is required to indicate the states where a reason is applicable. States may be configured to require status reasons, allow status reasons as optional or not allow status reasons. With this configuration, the framework will automatically get the list of valid reasons for a state that allows or requires them and then prompt the user for a status reason when a manual state transition occurs for that state. It also automatically triggers an error if the state requires a status reason and no reason is provided.

NOTE: The status reason configuration on the business object state is customizable. That means that for a product owned business object, an implementation may opt to change the delivered configuration.

Status reasons are defined for the parent (or "lifecycle") business object. All business objects in the hierarchy of the parent business object have the same valid reasons for their states.

The status reason code must be unique for the centralized status reason table. Business object and status are required fields, so it is not possible to share a common reason code (like "Not applicable") across multiple business objects or states. If multiple BOs / states want to support a reason "Not applicable" then each must define a unique record for it. This point should be considered when planning for your status reasons.

Selectable vs. Not Selectable

When defining a status reason, you may indicate whether it's **Selectable** or **Not Selectable**. When a manual transition is performed and a user is prompted for a status reason, only the **Selectable** reasons are presented. The **Not Selectable** reasons may be defined to support transitions that occur via algorithm processing.

NOTE: The Selectable setting is customizable. That means that if a product provides a base owned status reason for a business object state, an implementation may opt to change whether it is selectable or not. Careful consideration should be made before changing a base delivered status reason from **Not Selectable** to **Selectable** as this may affect base provided algorithm functionality that could be relying on the setting of **Not Selectable**.

Status Reason Business Object

The status reason maintenance object, as with many maintenance objects in the product, references a business object used to define attributes and behavior related to defining status reasons. The framework provides a business object for status reason (**F1-BOStatusReason**). For the business objects that have states that require a status reason (let's call these "transactional BOs"), if there is some special logic required for defining the status reasons, it is possible to define a different status reason BO. In this situation, the override status reason BO to use for capturing status reasons should be defined as a BO option on the transactional BO using the **Status Reason Business Object** option type. If a transactional BO does not define any status reason BO option, then the **F1-BOStatusReason** is used when adding a status reason.

Defining a Usage

The base product status reason BO provides the ability to define a "usage" value. This is useful for algorithms that perform state transitions where a status reason is needed and where the algorithm is usable by more than one business object. In this case, the status reason to use cannot be provided as a parameter because each business object must define its own set of status reasons for each state. The Usage value can be used instead. Each business object can configure the status reason to use in the algorithm and set the appropriate usage value. The algorithm can reference the usage value and retrieve the correct status reason to use based on the record's transactional BO.

The status reason business object provided with the framework product (**F1-BOStatusReason**) supports capturing a usage. The valid usage values are defined in the **Status Reason Usage** characteristic type.

Alternatives for Defining Reasons

There may be business objects in the system that capture reasons that are defined somewhere besides the BO status reason table. For example, some objects may have an explicit administrative table for status reasons. Some objects may use a Lookup or an Extendable Lookup to capture reasons. Refer to the business object description for information about how valid reasons are defined, if applicable.

If a business object supports a reason that is not related to a state transition (such as a creation reason), the BO status reason would not be used. One of the alternate methods for defining a reason, described above, would be used.

BO Algorithm Execution Summary

This table highlights the processing steps that occur when adding or changing a record that is governed by a business object.

Invoke BO	
Event	Comments
BO Pre-processing algorithms executed	These algorithms are only executed when Invoke BO is used. The business object in the Invoke BO is the one whose rules are executed.
MO Processing	
Event	Comments
Determine if status has changed.	The system keeps a note of the new status value but initially proceeds with the old value.

MO Processing.	Standard MO processing, including MO validation is executed.
Determine BO algorithm executed.	The MO level algorithm is executed to determine the identifying BO .
BO Validation algorithms executed.	
State transition rules are performed if the status has changed.	<p>BO Status Exit algorithms for the “old” status executed.</p> <p>Status updated to the new value.</p> <p>BO Status Enter algorithms for the “new” status executed.</p> <p>If no error — MO Transition algorithms are executed.</p> <p>FASTPATH: Refer to State Transitions are Audited for more information.</p> <hr/> <p>If error and there are “save points” the MO Transition Error algorithms are executed.</p> <p>FASTPATH: Refer to Keeping An Entity In Its Last Successful State for more information.</p> <hr/> <p>Otherwise, the error is reported.</p>
BO Status Monitor algorithms are executed.	If the record transitions again, the prior step (State transition rule step) is repeated for the new transition.
BO Post-processing algorithms are executed.	
BO Audit algorithms are executed.	These algorithms are only executed if the system detects a change in elements that are not marked with “no audit”.

NOTE: To emphasize, the steps in the MO Processing table are only executed when the maintenance object service is invoked. Any add or update initiated by an “invoke BO” statement will invoke the MO service. This is also true for web service that invoke the business object. The [Monitor Batch Process](#) does not invoke the maintenance service. By default the monitor batch process only executes the monitor algorithms and the state transition rules (if the monitor algorithms indicate that a status change should occur). However, it is possible for an Enter algorithm, Exit algorithm or Monitor algorithm to indicate that the other algorithms should be executed by the batch process. If the “force post processing” indicator is set to true, then the batch process invokes the BO Post Processing, BO Audit and MO Audit algorithms.

NOTE: For records that do not have a status, the state transition rules and the monitor rules are not applicable.

Granting Access To Business Objects

Every business object must reference an [application service](#). When you link a business object to an application service, you are granting all users in the group access to its instances. You can prevent users from transitioning a business object into specific states by correlating each business object status with each application service action (and then don't give the user group rights to the related action).

FASTPATH: Refer to [The Big Picture Of Application Security](#) for information about granting users access rights to an application service.

The system checks if a user has access rights each time the application is invoked to add, change, delete, read, or transition a business object. However, if a business object invokes another business object, we assume that access was controlled by the initial business object invocation and we do not check access for other business objects that it invokes. In other words, access rights are only checked for the initial business object invoked in a service call.

In order to apply business object security the system must be able to determine the business object associated with the actual object being processed. To do that the Maintenance Object itself has to have a **Determine BO** algorithm [plugged in](#). If this algorithm is not plugged in or it cannot determine the BO on the MO, the system will NOT invoke any BO rules. If the business object cannot be determined for a maintenance object instance, business object security is not checked. In this case the system checks the user's access rights using standard maintenance object security.

NOTE: Parent business objects are ignored. If a child business object exists, a user need only have access to the child business object's application service (not to every application service in the business object hierarchy).

Defining Business Objects

The topics in this section describe how to maintain business objects.

Note that several context sensitive dashboard zones appear on this page and are visible on all tabs.

- **Schema Tips.** This zone provides several links to launch help topics related to valid schema syntax and UI Hint syntax in one click.
- **View UI Rendering.** This zone provides buttons to view the automatic rendering of the Display map or Input map based on the attributes defined in the schema, including UI hints.
- **Generate Schema.** This zone includes a button that can be used to generate a “physical” schema based on the maintenance object definition. The element names are taken from the Java field name for each column. Once generated, adjust the schema as desired.
- **Create a BO Algorithm.** This zone includes a button to create a script based algorithm related to this business object. You are then prompted for information regarding the plug-in spot (BO or BO Status) and the system event, the name, description, etc. Once all the information is provided, the system creates an algorithm type, algorithm, links the algorithm to the business object, creates the script and brings you to the script step to start defining the logic for the plug-in script.
- **BOs Linked to the MO.** This zone displays other business objects for the same maintenance object as the BO currently displayed. You may drill into any of the other BOs by clicking its description.

Business Object - Main

Use this page to define basic information about a business object. Open this page using **Admin > System > Business Object**

Description of Page

Enter a unique **Business Object** name and **Description**. Use the **Detailed Description** to describe the purpose of this business object in detail. **Owner** indicates if this business object is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new business object, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Enter the **Maintenance Object** that is used to maintain objects of this type.

Enter a **Parent Business Object** from which to **inherit** business rules.

Lifecycle Business Object is only displayed for child business objects, i.e. those that reference a parent business object. It displays the highest-level business object in the inheritance hierarchy. Refer to [Inheriting Lifecycle](#) for more information.

Application Service is the application service that is used to provide security for the business object. Refer to [Granting Access To Business Objects](#) for more information. The application service on the child business object must have the same valid actions as the application service on the parent business object.

Use **Instance Control** to allow or prevent new entities from referencing the business object. Typically only the **identifying BOs** are marked to allow new instances.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the **schema viewer** window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Click the **View MO** hyperlink to view the maintenance object in the [Maintenance Object Viewer](#). You may find it useful to leave the application viewer window open while defining your business object schema.

The options grid allows you to configure the business object to support extensible options. Select the **Option Type** dropdown to define its **Value**. **Detailed Description** may display additional information on the option type. Set the **Sequence** to **1** unless the option can have more than one value. **Owner** indicates if this option is owned by the base package or by your implementation (**Customer Modification**).

NOTE: You can add new options types. Your implementation may want to add additional option types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do that, add your new values to the customizable lookup field **BUS_OBJ_OPT_FLG**. If you add a new option type for a business option, you must update its maintenance object to declare this new option type. Otherwise, it won't appear on the option type dropdown. You do that by referencing the new option type as a **Valid BO Option Type**[maintenance object option](#).

Where Used

Follow this link to open the data dictionary to view the tables that reference [F1_BUS_OBJ](#).

Business Object - Schema

Use this page to maintain a business object's schema. Open this page using **Admin > System > Business Object** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business object.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

The [Schema Designer](#) zone allows you to edit the business object's schema. The purpose of the schema is to describe the business object's properties and map them to corresponding maintenance object fields.

FASTPATH: Refer to [Schema Syntax](#) and [UI Hint syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the dashboard provides links to launch these help topics directly.

NOTE: Generating a Schema A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the fields for all the tables for the BO's maintenance object.

NOTE: View UI Rendering. A context sensitive "View UI Rendering" zone is associated with this page. The zone is useful for [business objects that define the user interface detail](#) using schema attributes and UI Hints. The buttons allow you to view the automatically rendered display and input maps.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Business Object - Algorithms

Use this page to maintain a business object's algorithms. Open this page using **Admin > System > Business Object** and then navigate to the **Algorithms** tab.

Description of Page

The **Algorithms** grid contains algorithms that control important functions for entities defined by this business object. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to [Plug-In Scripts](#) for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes each **System Event**. Refer to [A Business Object May Define Business Rules](#) for more information about these system events.

System Event	Optional / Required	Description
Audit	Optional	<p>Algorithms of this type may be used to audit certain changes made to business object instances.</p> <p>The system hands over to the algorithms a summary of all the elements that were changed throughout a specific call to update an object. Excluded from this processing are elements explicitly marked on the schema as requiring no audit. For each element its original value before the change as well as its new value are provided.</p> <p>It is the responsibility of the algorithms to record corresponding audit information.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Information	Optional	<p>We use the term "Business Object Information" to describe the basic information that appears throughout the system to describe an entity defined by the business object. The data that appears in this information description is constructed using this algorithm.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number found on the business object closest to the current business object in the inheritance hierarchy. Refer to Business Object inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>

System Event	Optional / Required	Description
Post-Processing	Optional	<p>Algorithms of this type may be used to perform additional business logic after a business object instance has been processed.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Pre-Processing	Optional	<p>Algorithms of this type further populates a request to maintain a business object instance right before it is processed.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Validation	Optional	<p>Algorithms of this type may be used to validate a business object instance when added, updated or deleted.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>

FASTPATH: Refer to [BO Algorithm Execution Summary](#) for more information about how these algorithms fit within the business object processing.

NOTE: Generate Algorithm. A context sensitive "Generate a BO Algorithm" zone is associated with this page. Refer to [Defining Business Objects](#) for more information about this zone.

NOTE: You can add new system events. Your implementation may want to add additional business object oriented system events. For example, your implementation may have plug-in driven logic that would benefit from a new system event. To do that, add your new values to the customizable lookup field **BO_SEVT_FLG**. If you add a new business object system event, you must update the maintenance object to declare this new system event. Otherwise, it won't appear on the system event dropdown. You do that by referencing the new system event as a **Valid BO System Event maintenance object option**.

NOTE: You can inactivate algorithms on base Business Objects. Your implementation may want to use a business object provided by the base product, but may want to inactivate one or more algorithms provided by the base business object. To do that, on the business object where this algorithm is referenced, go to the options grid on Business Object - Main and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

Business Object - Lifecycle

Use this page to maintain a business object's lifecycle oriented business rules and options. Open this page using **Admin > System > Business Object** and then navigate to the **Lifecycle** tab.

Description of Page

The **Status** accordion contains an entry for every status in the object's [lifecycle](#). The entry appears differently for a child business object as it can only extend its inherited lifecycle by introducing algorithms and options of its own.

Use **Status** to define the unique identifier of the status. This is NOT the status's description, it is simply the unique identifier used by the system. Only the highest-level business object can define lifecycle statuses. For a child business object the inherited status description is displayed allowing navigation to the corresponding entry on the business object defining the lifecycle.

Use **Description** to define the label of the status. This field is hidden for a child business object.

Use **Access Mode** to define the action associated with this status. Refer to [Access Rights](#) for the details of how to use this field to restrict which users can transition a business entity into this state. This field is hidden for a child business object.

Enter a **Monitor Process** to defer the monitoring of entities in this state until the specific batch process runs. Refer to [Monitor Rules](#) for more information. This field is hidden for a child business object.

The **Status Reason** dropdown indicates if users should be prompted to provide a specific reason when the business object enters this state. This field appears only if the Status Reason Field is configured as an option on the business object's maintenance object. Valid values are blank, **Optional**, and **Required**. The default value is blank (users are not prompted to provide a status reason). See [Configuring Status Reasons](#) for more information about status reasons.

Use **Status Condition** to define if this status is an **Initial**, **Interim** or **Final** state. Refer to [One Initial State and Multiple Final States](#) for more information about how this field is used. This field is hidden for a child business object.

Use **Transitory State** to indicate whether a business entity should ever exist in this state. Only **Initial** or **Interim** states can have a transitory state value of **No**. Refer to [transitory states](#) for more information. This field is hidden for a child business object.

Use **Alert Flag** to indicate that being in this state warrants an application alert. This may be used by custom logic to provide an alert to a user that entities exist in this state. This field is hidden for a child business object.

Use **Display Sequence** to define the relative order of this status for display purposes. For example when displayed on the status accordion and on the summary tab page. This field is hidden for a child business object.

Algorithms

The **Algorithms** grid contains algorithms that control important functions for a given status. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to [Plug-In Scripts](#) for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes each **System Event**.

System Event	Optional / Required	Description
Enter	Optional	Algorithms of this type apply business rules when a business object instance enters a given state. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object Inheritance for more information. Click here to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Exit	Optional	<p>Algorithms of this type apply business rules when a business object instance exits a given state.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object Inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Monitor	Optional	<p>Algorithms of this type monitor a business object instance while in a given state. Typically these are used to auto-transition it to another state.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object Inheritance for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>

FASTPATH: Refer to [BO Algorithm Execution Summary](#) for more information about how these algorithms fit within other business object algorithms.

NOTE: Generate Algorithm. A context sensitive "Generate a BO Algorithm" zone is associated with this page. Refer to [Defining Business Objects](#) for more information about this zone.

NOTE: You can inactivate status level algorithms on base Business Objects. Your implementation may want to use a business object provided by the base product, but may want to inactivate one or more of the status oriented algorithms provided by the base business object. To do that, on the business object and status where this algorithm is referenced, go to the options grid and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

Next Statuses

Use the **Next Statuses** grid to define the valid statuses a business entity can transition into while it's in this state. This section is hidden for a child business object. Refer to [Valid States versus State Transition Rules](#) for more information. Please note the following about this grid:

- **Status** shows the statuses for the top-level business object, the **Status Code**, the **Lifecycle BO description**, and the **Status description** for each status.
- Use **Action Label** to indicate the verbiage to display on the button used to transition to this status.
- **Sequence** controls the relative order of one status compared to others for display purposes. This information may be used to control the order in which buttons are presented on a user interface.
- **Default** controls which next state (if any) is the default one. This information may be used by an **Enter** or **Monitor** algorithm to determine an auto-transition to the default state. It may also be used to also mark the associated button as the default one on a user interface.
- **Transition Condition** may be configured to identify a common transition path from the current state. By associating a given "next status" with a transition condition value, you can design your auto-transition rules to utilize those flag values without specifying a status particular to a given business object. Thus, similar logic may be used across a range of business objects to transition a business entity into, for example, the next **Ok** state for its current state. You'll need to add your values to the customizable lookup field **BO_TR_COND_FLG**.
- **Transition Role** controls whether only the **System** or both **System and User** have the ability to transition a business entity into a given "next status".

- When you initially set up a business object lifecycle, none of the statuses will reside on the database and therefore you can't use the search to define a "next status". We recommend working as follows to facilitate the definition of this information:
 - Leave the Next Statuses grid blank when you initially define a business object's statuses
 - After all statuses have been saved on the database, update each status to define its Next Statuses (this way, you can use the search to select the status).

Options

The options grid allows you to configure the business object status to support extensible options. Select the **Option Type** drop-down to define its **Value**. **Detailed Description** may display additional information on the option type. Set the **Sequence** to **1** unless the option can have more than one value. **Owner** indicates if this option is owned by the base package or by your implementation (**Customer Modification**).

NOTE: You can add new options types. Your implementation may want to add additional option types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do that, add your new values to the customizable lookup field **BO_OPT_FLG**. If you add a new option type for a status, you must update the business object's maintenance object to declare this new option type. Otherwise, it won't appear on the option type dropdown. You do that by referencing the new option type as a **Valid BO Status Option Type** [maintenance object option](#).

Business Object - Summary

This page summarizes business object information in a high level. Open this page using **Admin > System > Business Object > Search** and then navigate to the **Summary** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business object.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

The **Business Object Hierarchy** zone displays in a tree view format the [hierarchy](#) of child business object associated with the current business object. It also shows the current business object's immediate parent business object.

For business objects with a lifecycle, the **Lifecycle Display** zone shows a graphical depiction of the lifecycle. Refer to the inline help of that zone for more information.

The **Options** zone summarizes business object and state specific options throughout the [inheritance](#) chain.

The **Rules** zone summarizes business object and state specific rules throughout the [inheritance](#) chain.

Advanced BO Tips and Techniques

The topics in this section describe some advanced tips and techniques for configuring business objects.

Managing To Do Entries

The product provides several base algorithm types that may be used to manage To Do entries through status changes for a given record via BO lifecycle plug-ins.

Create To Do Entry

The product supplies a BO status Enter algorithm type **Generic To Do Creation (F1-TDCREATE)** that creates a To Do entry based on parameter configuration. Refer to the algorithm type description for more information about how it determines the To Do type or To Do role and how to populate the appropriate message text onto the To Do. This algorithm may be used in conjunction with the Retry Logic (below).

If your implementation has a business rule that requires a To Do entry to be created when entering a given BO status and the logic provided by the algorithm type meets the needs of the business rule, this algorithm type may be used. Create an algorithm for the algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as an Enter algorithm.

Retry Logic

The algorithm type **Retry for To Dos (F1-TODORETRY)** is supplied for a special use case. It is a BO status monitor plugin and may be used for a state that is a type of ‘error’ or ‘waiting’ state. It relies on the To Do entry creation logic to set a Retry Frequency. The algorithm transitions to the originating state to retry the logic. The idea is that the condition that caused the record to enter the ‘error’ or ‘waiting’ state may be resolved after some period of time has passed, allowing the record to progress in its lifecycle. Refer to the algorithm type description for more information about its logic.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as a Monitor algorithm. The state should also have an algorithm for the **Generic To Do Creation** algorithm type plugged in as an Enter algorithm (or something equivalent) that sets the appropriate Retry Frequency.

To Do Completion

It is common that one or more To Do entries associate with a given record should be completed when exiting a state (if it is not already completed). The system supplies the algorithm type **Generic To Do Completion (F1-TODOCOMPL)** that may be used for this purpose. Note that the algorithm type functionality is not tied to any To Do creation logic. It may be used for any use case where To Do entries should be completed on exiting a state. Refer to the algorithm type description for more information about its functionality and how to prevent certain To Do entries from being automatically completed.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as an Exit algorithm.

Submitting a Batch Job

The product provides a base algorithm type that submits a batch job when entering a BO state. This functionality allows for “event driven” batch submission where the event is the lifecycle transition for a certain record.

The algorithm type is **Create Batch Job Submission Entry for Batch Control (F1-SCHEDJOB)**. The batch control code is a parameter for the algorithm. Refer to the algorithm type description for more information about its logic.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameter with the batch control that should be submitted and plug the new algorithm into the appropriate business object status as an Enter algorithm.

Defining Status Reasons

Status Reasons are used to provide more information about why a business object transitioned to a given state. The status reason table provides a centralized place where status reasons can be defined across many different business objects and states.

NOTE: Refer to [Defining Reasons for Entering a State](#) for overview information.

If a business object has one or more states that are configured to capture a status reason, you may configure the valid reasons by navigating to the status reason portal using **Admin > System > Status Reason**.

The topics in this section describe the base-package zones that appear on the Status Reason portal.

Business Objects with Status Reason List

This zone displays the business objects that have one or more status values that allow status reasons to be defined.

Click the broadcast icon to open other zones that contain more information about the business object's status reasons.

Status Reasons

The **Status Reasons** zone contains a list of the existing status reasons for the broadcasted business object.

Business Services

A business service is used to expose a back-end service so that it may be invoked by a script or a zone or a map to retrieve information or perform functions, depending on the related service.

As with the business object, the business service's interface to the internal service is defined using its schema. The schema maps the business service's elements to the corresponding elements in the internal service program's XML document. Just as a business object can simplify the schema of its maintenance object by only defining elements that it needs and “flattening” entries in a child collection to be defined as a singular element, a business service schema may simplify its service XML in a similar way.

FASTPATH: Refer to [Schema Syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema.

[Inbound web services](#) and [scripts](#) support interaction with business services. You can also invoke a business service from a Java class.

Service Program

This transaction defines services available in the system. These include user interface services as well as stand-alone services that perform a specific function. A service may be referenced by a business service. Use this transaction to view existing service and introduce a new stand-alone service to be made available to a Business Service.

Select **Admin > System > Service Program** to maintain service programs.

Description of Page

Service Name is the unique identifier of the service.

CAUTION: Important! When adding new service programs, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Owner indicates if this service is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a service. This information is display-only.

Description describes the service.

Service Type indicates whether the service is a **Java Based Service** or a **Java (Converted) Service**.

This **Program Component** grid shows the list of program user interface components associated with the service. For a stand-alone service, this list is typically not applicable.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MD_SVC](#).

Defining Business Services

The topics in this section describe how to maintain business services.

Note that several context sensitive dashboard zones appear on this page and are visible on all tabs.

- **Schema Tips.** This zone provides several links to launch help topics related to valid schema syntax.
- **Generate Schema.** This zone includes a button that can be used to generate the schema based on the XML of its related Service. Once generated, adjust the schema as desired.

Business Service - Main

Use this page to define basic information about a Business Service. Open this page using **Admin > System > Business Service**

Description of Page

Enter a unique **Business Service** name and **Description**. Use the **Detailed Description** to describe the purpose of this business service in detail. **Owner** indicates if the business service is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new business service, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Enter the internal **Service Name** being called when this business service is invoked.

Enter the **Application Service** that is used to provide security for the business service. The application service must have an Access Mode of Execute.

Click the **View Schema** to view the business service's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Click the **View XML** hyperlink to view the XML document used to pass data to and from the service in the [Service XML Viewer](#). You may find it useful to leave the application viewer window open while defining your business service schema.

NOTE: XML document may not be viewable. If you create a new service program and do not regenerate the application viewer, you will not be able to view its XML document.

Where Used

Follow this link to open the data dictionary to view the tables that reference [FI_BUS_SVC](#).

Business Service - Schema

Use this page to maintain a Business Service's schema and to see where the Business Service is used in the system. Open this page using **Admin > System > Business Service** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business service.

The [Schema Designer](#) zone allows you to edit the business service's schema. The purpose of the schema is to map the business service's elements to the corresponding fields of the back-end service program it rides on.

NOTE: Generating a Schema A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the elements that are found in the XML of the BS's service.

FASTPATH: Refer to [Schema Syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the dashboard provides a link to launch this help topic directly.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Useful Services and Business Services

The following section highlights some business services and services provided by the product that may be useful for implementations to use.

Data Explorer Service

The system provides a mechanism for performing an SQL select statement for use in scripting, Java plug-ins, or via a web service call. This is done by creating a zone using one of the data explorer zone types where the SQL is defined. Then, create a business service using the Data Explorer service (**FWLZDEXP**).

NOTE: There are numerous business services delivered with the base product that reference this service that may be used as a template.

The following points highlight how to create your own business service for this service. Note that typically a separate business service exists for each zone.

- Enter a **Business Service** code and a **Description**. It is recommended to define the business service code to match the zone code so that it's easier to manage which business service invokes which zone.
- Select the **Service Name**FWLZDEXP.
- On the **Schema** tab, under the `<schema>` node, enter mapping for the fields that are required for the Data Explorer service:
 - The **Zone** should be mapped into service field `ZONE_CD` . Define the zone code as the default value.
 - For every **user filter** defined on the zone, create a schema mapping into the service field `Fx_VALUE`, where "x" is the filter number (from the zone parameters).
 - For every **hidden filter** defined on the zone, create a mapping into the service field `Hx_VALUE`, where "x" is the filter number (from the zone parameters).
 - The search results are returned as a list by the data explorer service. Each column value is in the service field `COL_VALUE` with an appropriate sequence number (`SEQNO`). The results can be [flattened](#) based on sequence number allowing for a logical element name to be defined.
 - Another useful field is `ROW_CNT`, which provides the number of rows retrieved by your search.

The following is an example of the schema for a BS that receives a business object code and returns a list of status values and their descriptions that allow status reasons to be defined.

```
<schema>
  <zone mapField="ZONE_CD" default="F1-BOSTSLST" />
  <bo mapField="H1_VALUE" />>
  <rowCount mapField="ROW_CNT" />>
```

```

<results type="list" mapList="DE">>
  <status dataType="string" mapField="COL_VALUE">
    <row mapList="DE_VAL">>
      <SEQNO is="1" />>
    </row>>
  </status>>
  <description dataType="string" mapField="COL_VALUE">
    <row mapList="DE_VAL">>
      <SEQNO is="2" />>
    </row>>
  </description>>
</results>>
</schema>

```

Maintenance Object Log Service

Many maintenance objects support a log table that follows a pattern of column names and behavior. The system provides a service called Generic MO Log Service (**F1MOLOGP**) that may be used to perform common functions related to log entries:

- Read log entries. If you pass a certain MO, primary key and log sequence number, the service will return the details of that log entry. The product provides a generic business service that may be used for this purpose — Generic MO - Retrieve Log Details (**F1-ReadMOLog**). Alternatively, it is possible to create a business service for a given MO where the MO code is assigned to the MO element using the default syntax. This allows business functionality specific to that maintenance object to use the specific BS.
- Add log. The service may be used to add a log entry. If a user log is added, then the comments from the user are populated in the detailed description. System generated log entries typically supply a message category / message number along with other information such as the status, a specific log type and optionally a related object reference (via a characteristic). The product provides a generic business service that may be used for this purpose — Add Generic MO Log (**F1-AddMOLog**). Alternatively, it is possible to create a business service for a given MO where the MO code is assigned to the MO element using the default syntax. This allows business functionality specific to that maintenance object to use the specific BS.

Base Business Services

The following table highlights some business services provided by the product that may be useful for custom logic for an implementation.

CAUTION: This is not intended to be a complete reference of Business Services. Refer to the business service page to find all the supported business services.

Business Object Related Services

Business Service Name	Description
F1-AutoTransitionBO	Performs monitoring algorithms associated with the current state of a given business object instance (which may result in subsequent state transitioning).
F1-CompareBusinessObjectData	Compares two versions of a given business object instance.
F1-DetermineBo	Determines the business object of a given instance of a maintenance object by executing the MO's Determine BO logic.
F1-GetRequiredFieldsForBOState	Returns the required fields for a given business object status.
F1-RetrieveBOOption	Returns BO option values for a given BO and option type.
F1-RetrieveBOStatusOption	Returns BO option values for a given BO, status and option type.
F1-RetrieveBOStatusOption	Retrieves a list of BOs for a given MO that are accessible for the current user.

Business Service Name	Description
F1-RetrieveBoStatusDescription	Return the description of a given BO status.
F1-RetrieveBusinessObjectLabel	Return the label appropriate for a given path (e.g. element) within a BO schema.
F1-RetrieveNextStates	Return a list of next possible states based on the input of a MO and its prime key, or a BO and one of its statuses.

Email Related Services

Business Service Name	Description
F1-EmailService	Sends an email message in real time.
F1-RetrieveEmailAddress	Retrieves the email addresses of users belonging to a To Do Role.
F1-RetrieveEnvironmentURL	Retrieves the current environment URL information for the installation.

Tools for Maps and Scripting

Business Service Name	Description
F1-DateMath	Performs various date and time math calculations. Refer to the BS description for more details.
F1-DateTimeFormattingService	Formats a given date / time based on the user's display profile settings.
F1-ExecuteScriptInNewSession	Executes a Service Script in a new processing session/transaction.
F1-GetFieldLabel	Retrieves the label for a given field.
F1-GetForeignKeyReference	Returns foreign key reference information for a given FK Reference and primary key, including info description, navigation option, and context menu.
F1-GetFKReferenceDetails	Returns foreign key reference information for a given MO and primary key, including FK reference code, info description, navigation option, search zone and context menu.
F1-GetLookupDescription	Returns lookup description for a lookup field value given the lookup field name.
F1-GetExtLookUpVal	Returns the list of values for a given extendable lookup BO.
F1-GetMonthInYearAbbreviation	Returns a 3-character month abbreviation for an input date in system format.
F1-NumberAmountFormatter	Formats a given amount or number based on the user's display profile settings. It also may receive input to adjust the scale and optionally apply currency settings.
F1-OutmsgDispatcher	Dispatches a real-time message giving the user the option of whether to persist the message on the database, and whether to trap errors that may take place during the call.
F1-RethrowError	Issues an application error using the input message category / number / parameters.
F1-RetrieveMODescription	Retrieves the description for a maintenance object.

Business Service Name	Description
F1-ReturnMessage	Returns the expanded message given a message category, number, parameters, and parameter types.
F1-SavePointDispatcher	Allows for a service script to be executed where exceptions are trapped and the transaction is rolled back to a save point set before the service script execution.

User Related Services

Business Service Name	Comments
F1-CheckApplicationSecurity	Checks a user's security for a given application service / access mode
F1-CheckUserAuthorization	Determine whether a given user is authorized for access based on the input application service, security code, and authorization level.
F1-DeterminelfUserCanApproveTD	Determine if the current user can approve a given To Do.

User Interface (UI) Maps

The User Interface (UI) map holds HTML to be rendered within [portal zones](#) and [Business Process Assistant \(BPA\) scripts](#). UI maps allow your implementation to create input forms and output maps that closely match your customer's business practices. In other words, the UI Map is designed to facilitate the capture and display of your [business objects](#) and [business services](#).

The UI map is a repository for a single HTML document paired with an XML schema where the schema defines the data that the HTML document displays and/or modifies. The UI Map HTML gives you the ability to craft the display by any method that an html document can support, including JavaScript and full CSS functionality.

Configuration tool support for UI Maps hinges around the ability to inject and extract an XML document from the HTML. For more information on the specialized support for HTML and JavaScript functionality refer to [UI Map Attributes and Functions](#).

```

<html>
<head>
<title>Output Personal Information</title>
<link rel="stylesheet" type="text/css" href="cm_templates/cmStyles.css"/>
</head>
<body>

<table width="100%" border="0" cellpadding="2" style="margin-top:15px;" >

  <tr>
    <td/>
    <td/> <!-- locate the edit button on the bottom of the third column -->
    <td rowspan="99" style="vertical-align:bottom; margin-left:5px">
      <input type="button" value="edit" onClick="oraRunScript('HumanInfoU','personId');"/>
    </td>
  </tr>

  <tr>
    <td class="outputLabel">Name:</td>
    <td><span oraField="name" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Home Phone:</td>
    <td><span oraField="homePhone" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Business Phone:</td>
    <td><span oraField="businessPhone" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Cell Phone:</td>
    <td><span oraField="cellPhone" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">FAX:</td>
    <td><span oraField="fax" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Social Security:</td>
    <td><span oraField="socialSecurity" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Drivers License:</td>
    <td><span oraField="driversLicense" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Email:</td>
    <td><span oraField="email" class="outputData"></span></td>
  </tr>

</table>

</body>

<xml>
<root>
  <name>Greer, Johan</name>
  <email>jurgen.greer@media.com</email>
  <socialSecurity>939-30-3939</socialSecurity>
  <driversLicense>C8392020</driversLicense>
  <homePhone>(838) 030-0303</homePhone>
  <cellPhone>(444) 444-4040</cellPhone>
  <businessPhone>(737) 393-3838</businessPhone>
  <fax>(373) 939-3939</fax>
  <personId>1239997654</personId>
</root>
</xml>
</html>

```

Figure 1: HTML to Display Customer Business Object

Name:	Greer, Johan
Home Phone:	(838) 030-0303
Business Phone:	(737) 393-3838
Cell Phone:	(444) 444-4040
FAX:	(373) 939-3939
Social Security:	939-30-3939
Drivers License:	C8392020
Email:	jurgen.greer@media.com
	<input type="button" value="edit"/>

Figure 2: Customer HTML Rendered (Output Data for Zone)

UI maps are typically crafted as output tables when used in conjunction with portal zones - please refer to [Map Zones](#) for more information. When referenced within [BPA scripts](#), UI maps are typically crafted as forms for the capture and update of data.



Figure 3: HTML Input Form Rendered (for BPA Script)

Portal zones can reference a UI map for the zone header. They may also utilize a UI map to define their filter area. This type of UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

NOTE: UI Map Tips. A context sensitive "UI Map Tips" zone is visible on the UI map maintenance page. This zone provides several links to launch help topics related to valid schema syntax and UI Hint syntax in one click.

NOTE: Editing HTML. You can use a variety of HTML editors to compose your HTML, which you can then cut, and paste into your UI map. In addition, the zone provides a complete list of the XML schema nodes and attributes available to you when you construct the map's data schema.

Defining UI Maps

The topics in this section describe how to maintain UI Maps.

UI Map - Main

Use this page to define basic information about a user interface (UI) Map. Open this page using **Admin > System > UI Map**

Description of Page

Enter a unique **Map** name. **Owner** indicates if the UI map is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new UI map, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Use **UI Map Type** to indicate whether the map is a **Complete HTML Document** or an **HTML Fragment**. Portal zones can reference a UI map to describe a fragment of their HTML, for example the zone header or filter area. In this case the UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

Enter a **Description**. Use the **Detailed Description** to describe how this map is used in detail.

Click on the **View Schema** to view the UI map's expanded schema definition. Doing this opens the [schema viewer](#) window.

Use the **Test UI Map** hyperlink to render your html in a test window.

NOTE: The **Test UI Map** hyperlink also exercises the proprietary functionality that binds an xml element with an html element so you can get immediate feedback on your html syntax.

Where Used

Follow this link to open the data dictionary to view the tables that reference [FI_MAP](#).

UI Map - Schema

Use this page to maintain a UI Map's HTML and schema and to see where the UI Map is used in the system. Open this page using **Admin > System > UI Map** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the UI Map.

The **HTML Editor** zone allows you to edit the HTML document of the map.

NOTE: Refer to [UI Map Attributes and Functions](#) and [UI Map Standards](#) for more information about HTML definition syntax. These topics describe good ways to produce simple HTML, however, they are not an HTML reference. Note that you can use a variety of HTML editors to compose your HTML, which you can then cut and paste into your UI map.

NOTE: Providing Help. A [tool tip](#) can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map. Refer to [UI Map Attributes and Functions](#) for more information on how to enable and provide UI map help.

The **Schema Designer** zone allows you to edit the data schema part of the map. The purpose of the schema is to describe the data elements being displayed by the map.

FASTPATH: Refer to [Schema Syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **UI Map Tips** zone in the dashboard provides a link to launch this help topic directly.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

UI Map Attributes and Functions

Contents

- [Bind XML to HTML](#)
- [Build a Dropdown List](#)
- [Format Input and Output Fields](#)
- [Search Using a Pop-Up Explorer Zone](#)
- [Enable UI Map Help](#)
- [Display Labels](#)
- [Display Errors](#)
- [Fire JavaScript for Browser Events](#)
- [Hide Elements](#)
- [Invoke Schema Based Services](#)
- [Refresh a Rendered Map or Portal Page](#)
- [Embed Framework Navigation](#)
- [Launch BPA Script](#)
- [Exit UI Map with Bound Values](#)
- [Include a Map Fragment](#)
- [Show Schema Default on Add](#)
- [Configure a Chart](#)
- [Upload and Download a CSV File](#)
- [Construct Portal Zone Map Fragments](#)
- [Required JavaScript Libraries](#)

Bind XML to HTML

Only two different attributes are required to bind a UI Map's XML to its HTML. Both of these attributes require an XML document embedded within the HTML, where the XML is bounded by <xml> nodes.

WARNING: You must embed a pair of <xml></xml> tags within your HTML document for binding to occur.

oraField="field element xpath"

The oraField attribute is used to link an HTML element directly with an XML element, where the XML element is defined within the UI Map's XML schema. The oraField attribute can be used with any *rendering* HTML element, such as: , <div>, and <input>.

- HTML for input element:

```
<html>
<body>
<table>
  <tr>
    <td>Address:</td>
    <td><input type="text" oraField="address" /></td>
  </tr>
  <tr>
    <td>City:</td>
    <td><input type="text" oraField="city" /></td>
  </tr>
  <tr>
    <td>State:</td>
    <td><input type="text" oraField="state" /></td>
  </tr>
  <tr>
    <td>Zip:</td>
    <td><input type="text" oraField="zip" /></td>
  </tr>
</table>
</body>
```

```

<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

HTML for input element rendered:

Address:	<input type="text" value="123 Main St"/>
City:	<input type="text" value="Alameda"/>
State:	<input type="text" value="CA"/>
Zip:	<input type="text" value="94770"/>

Figure 4: HTML input elements rendered

- HTML for span and div elements:

```

<html>
<body>

<div oraField="address"></div>
<span oraField="city"></span>
<span>,</span>
<span oraField="state"></span>
<span oraField="zip"></span>
<span oraField="country"></span>

</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

HTML for span and div elements rendered:

123 Main St
Alameda, CA 94770

Figure 5: HTML span elements rendered

oraList="list element xpath"

The oraList attribute is used to link an HTML table with an XML list, where the XML list is defined within the UI Map's XML schema. The purpose of the oraList element is to trigger the framework to replicate the table's HTML for each occurrence of the list.

NOTE: The oraField attributes embedded within the list must contain XPath navigation relative to the list. See below for an example.

```

<html>
<head><title>Bind xml list element</title></head>
<body>
<table oraList="payment">
  <thead>
    <tr>

```

```

        <th><span>Pay Date</span></th>
<th><span>Amount</span></th>
    </tr>
    </thead>
    <tr>
        <td>
            <span oraField="date" oraType="date"></span>
        </td>
        <td align="right">
            <span oraField="amount" oraType="money"></span>
        </td>
    </tr>
</table>
</body>
<xml>
<root>
    <payment>
        <date>2008-01-01</date>
        <amount>44.28</amount>
    </payment>
    <payment>
        <date>2008-02-01</date>
        <amount>32.87</amount>
    </payment>
    <payment>
        <date>2008-03-01</date>
        <amount>21.76</amount>
    </payment>
</root>
</xml>
</html>

```

HTML for XML list rendered:

Pay Date	Amount
01-01-2008	\$44.28
02-01-2008	\$32.87
03-01-2008	\$21.76

Figure 6: XML list rendered in HTML

Build a Dropdown List

The following attributes are provided to build an HTML 'select' element, also called a dropdown, based on various sources.

- Lookup. **oraSelect="lookup:LOOKUP_FIELD;"** Example:

```

...
<td>House Type:</td>
<td>
    <select oraField="houseType" oraSelect="lookup:HOUSE_TYPE;"></select>
</td>

```

- Extendable Lookup. **oraSelect="lookupBO:BUS_OBJ_CD;"** Example:

```

...
<td>UI Device Display Type:</td>
<td>
    <select oraField="uiDeviceType" oraSelect="lookupBO:F1-DeviceDisplayTypes;"></select>
</td>

```

- Characteristic Type (pre-defined). **oraSelect="charType:CHAR_TYPE_CD;"** Example:

```

...
<td>Usage:</td>
<td>
    <select oraField="statusReasonUsage" oraSelect="charType:F1-SRUSG;"></select>

```

```
</td>
...
```

- Control Table **oraSelect="table:TABLE_NAME;"** Example:

```
...
<td>Currency: </td>
<td>
  <select oraField="currency" oraSelect="table:CI_CURRENCY_CD;"></select>
</td>
...
```

NOTE: This attribute only works with tables that follow the standard control table structure where there is a related language table that includes the column **DESCR** as its description column. You can use the Application Viewer data dictionary to identify tables that qualify for this functionality.

CAUTION: The oraSelect function will only work if less than 500 values are displayed. Only use this function if the table has less than 500 values.

- Output from a Service Script or a Business Service that returns a list of value / description pairs. For this option, there is syntax to support defining one or more values to pass into the service. In addition, there is syntax to identify the output values / descriptions.
 - The **oraSelectIn** is used to pass one or more values into the service. For every input parameter, first list the path of the element in the service, then indicate the value you are passing from the map (separated by a colon). The value being passed from the map may be a literal value or it may be the XPath of another element.
 - The **oraSelectOut** is used to identify the output elements from the service that return the value and description pairs.

The syntax is as follows:

- oraSelect="ss:service script" oraSelectIn="servicePath:element | 'literal';"**
oraSelectOut="valuePath:servicePath; descPath:servicePath"
- oraSelect="bs:business service" oraSelectIn="servicePath:element | 'literal';"**
oraSelectOut="valuePath:servicePath; descPath:servicePath"

Example using a business service:

```
...
<td>External System: </td>
<td>
  <select oraField="externalSystem" oraSelect="bs:F1-
RetrieveExternalSystems" oraSelectIn="outboundMsgType:boGroup/parameters/
outboundMsgType;" oraSelectOut="valuePath:results/externalSystem; descPath:results/
description"></select>
</td>
...
```

This method for building dropdowns is often used when there is a dependency between elements and the list of valid values in a dropdown (for the child element) is based on another element in the map (the parent element). When the parent element is changed, it may be required to refresh the child element. This behavior can be implemented using the function called within an **onChange** event in the map. The syntax is **oraHandleDependentElements('dependent element');**. Multiple target elements (dependents) can be named.

The following example is related to the above business service example where the list of external systems is specific for a given outbound message type, which is passed in as input. The snippet below shows the configuration for the outbound message type element to trigger a refresh of the external system's dropdown list.

```
...
<div>
  <label oraLabel="boGroup/parameters/outboundMsgType"></label>
  <span>
    <select oraSelect="table:F1_OUTMSG_TYPE" oraField="boGroup/parameters/
outboundMsgType" onChange="oraHandleDependentElements('boGroup/parameters/externalSystem');"></
select>

```

```

    </span>
</div>
...

```

- Output from a page service. For this option there is syntax to support build a select dropdown from a page service. The syntax is **oraSelect="service:service name;"**.

```

...
<td>Country:</td>
<td>
  <select oraField="country" oraSelect="service:CIPTCNTW;"></select>
</td>
...

```

- Embedded List. This option is used to build a select dropdown based on a list within the map's XML. The syntax is **oraSelect="valuePath:xpath;descPath:xpath"**.

```

<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Select: </td>
    <td><select oraSelect="valuePath:list/value; descPath:list/desc" oraField="target"></
select></td>
  </tr>
</table>
</body>
</html>
<root>
<target>10</target>
  <list>
    <value>10</value>
    <desc>Ten</desc>
  </list>
  <list>
    <value>20</value>
    <desc>Twenty</desc>
  </list>
  <list>
    <value>40</value>
    <desc>Forty</desc>
  </list>
</root>
</xml>
</html>

```

Format Input and Output Fields

The following attributes are designed to apply data type formatting for input and output fields.

oraSchemaDataTypes="false"

The oraSchemaDataTypes attribute is used to trigger automatic formatting in the rendered HTML document. The automated formatting will occur according to the dataType attributes defined in the UI map's schema. For details on specific data type formatting, please refer to the oraType attribute descriptions below.

WARNING: The attribute **oraSchemaDataTypes="true"** will be automatically injected into the UI map's HTML!

If you do not wish to apply the schema's data types to the rendered HTML then you must specify this attribute in the body node with a value of false. The attribute **<body oraSchemaDataTypes="false">** is required to avoid automatic formatting!

- UI Map schema:

```

<schema>
  <schemaDate dataType="date" />
  <schemaDateTime dataType="dateTime" />
  <schemaFKRef fkRef="CI_USER" />
  <schemaLookup dataType="lookup" lookup="ACCESS_MODE" />
  <schemaMoney dataType="money" />

```

```

    <schemaNumber dataType="number" />
    <schemaTime dataType="time" />
</schema>

```

- UI Map HTML:

```

<html>
<body oraSchemaDataTypes="true">
<table border="1" cellpadding="1" cellspacing="1">
<tr><th>dataType</th><th>result type</th><th>input result</th><th> display-only result</th></tr>

    <tr>
        <td rowspan="2">date (from schema)</td>
<td>raw</td>
<td><input oraField="schemaDate" oraType="string" /></td>
<td><span oraField="schemaDate" oraType="string"></span></td>
    </tr>
    <tr>
<td>rendered</td>
        <td><input oraField="schemaDate"></td>
<td><span oraField="schemaDate"></span></td>
    </tr>

    <tr>
        <td rowspan="2">dateTime (from schema)</td>
<td>raw</td>
<td><input oraField="schemaDateTime" oraType="string"></td>
        <td><span oraField="schemaDateTime" oraType="string"></span></td>
    </tr>
    <tr>
<td>rendered</td>
        <td><input oraField="schemaDateTime"></td>
<td><span oraField="schemaDateTime"></span></td>
    </tr>

    <tr>
        <td rowspan="2">fkRef (from schema)**</td>
<td>raw</td>
<td><input oraField="schemaFkRef" oraType="string"></td>
<td><span oraField="schemaFkRef" oraType="string"></span></td>
    </tr>
    <tr>
<td>rendered</td>
        <td><input oraField="schemaFkRef"></td>
<td><span oraField="schemaFkRef"></span></td>
    </tr>

    <tr>
        <td rowspan="2">lookup (from schema)</td>
<td>raw</td>
<td><input oraField="schemaLookup" oraType="string"></td>
<td><span oraField="schemaLookup" oraType="string"></span></td>
    </tr>
    <tr>
<td>rendered</td>
        <td><input oraField="schemaLookup"></td>
<td><span oraField="schemaLookup"></span></td>
    </tr>

    <tr>
        <td rowspan="2">money (from schema)</td>
<td>raw</td>
<td><input oraField="schemaMoney" oraType="string"></td>
<td><span oraField="schemaMoney" oraType="string"></span></td>
    </tr>
    <tr>
<td>rendered</td>
        <td><input oraField="schemaMoney"></td>
<td><span oraField="schemaMoney"></span></td>
    </tr>

```

```

    <tr>
      <td rowspan="2">number (from schema)</td>
    <td>raw</td>
    <td><input oraField="schemaNumber" oraType="string"/></td>
    <td><span oraField="schemaNumber" oraType="string"></span></td>
  </tr>
  <tr>
    <td>rendered</td>
    <td><input oraField="schemaNumber"></td>
    <td><span oraField="schemaNumber"></span></td>
  </tr>

  <tr>
    <td rowspan="2">time (from schema)</td>
  <td>raw</td>
  <td><input oraField="schemaTime" oraType="string"></span></td>
  <td><span oraField="schemaTime" oraType="string"></span></td>
  </tr>
  <tr>
    <td>rendered</td>
    <td><input oraField="schemaTime"></td>
    <td><span oraField="schemaTime"></span></td>
  </tr>

</table>

</body>
<xml>
<root>
<schemaDate>2007-11-02</schemaDate>
<schemaDateTime>2007-11-02-23.45.00</schemaDateTime>
<schemaFkRef>USD</schemaFkRef>
<schemaLookup>A</schemaLookup>
<schemaMoney>1000000</schemaMoney>
<schemaNumber>5661976.11548</schemaNumber>
<schemaTime>23.45.00</schemaTime>
</root>
</xml>
</html>

```

Is rendered as:

dataType	result type	input result		display-only result
date	raw	<input type="text" value="2007-11-02"/>		2007-11-02
	rendered	<input type="text" value="11-02-2007"/>		11-02-2007
dateTime	raw	<input type="text" value="2007-11-02-23.45.00"/>		2007-11-02-23.45.00
	rendered	<input type="text" value="11-02-2007"/>	<input type="text" value="11:45PM"/>	11-02-2007 11:45PM
fkRef	raw	<input type="text" value="SYSUSER"/>		SYSUSER
	rendered	<input type="text" value="SYSUSER"/>		SYSUSER
lookup	raw	<input type="text" value="A"/>		A
	rendered	<input type="text" value="Add"/>		Add
money	raw	<input type="text" value="1000000"/>		1000000
	rendered	<input type="text" value="\$1,000,000.00"/>		\$1,000,000.00
number	raw	<input type="text" value="5661976.11548"/>		5661976.11548
	rendered	<input type="text" value="5,661,976.11548"/>		5,661,976.11548
time	raw	<input type="text" value="23.45.00"/>		23.45.00
	rendered	<input type="text" value="11:45PM"/>		11:45PM

Figure 7: oraSchemaDataTypes="yes" rendered

oraType="date"

The oraType: date function is used to display a date according to the user's display profile.

NOTE: If you use oraType="date" within an <input> element it will be processed by the framework. For example, if you enter '1.1.00', '01-01-2000' will be defaulted when you tab out of the input field.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date: </td>
    <td><span oraField="date" oraType="date"></span></td>
  </tr>
  <tr>
    <td>Date: </td>
    <td><input oraField="date" oraType="date" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<date>2008-12-28</date>
</root>
</xml>
</html>
```

Is rendered as:

Date: 12-28-2008

Date:

Figure 8: oraType="date" rendered as output and input

oraType="time"

The oraType: time function is used to display a time according to the user's display profile.

NOTE: If you use oraType="time" within an <input> element it will be processed by the framework. For example, if you enter '12:28', '12:28AM' will be defaulted when you tab out of the input field.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Time: </td>
    <td><span oraField="time" oraType="time"></span></td>
  </tr>
  <tr>
    <td>Time: </td>
    <td><input oraField="time" oraType="time" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<time>00.28.54.389</time>
</root>
</xml>
</html>
```

Is rendered as:

Time: 12:28AM

Time:

Figure 9: oraType="time" rendered as output and input

oraType="dateTime; time:suppress"

The oraType: dateTime function is used to display a timestamp according to the user's display profile. Optionally, the time portion of the date time element can be suppressed.

NOTE: If you use oraType="dateTime" within an <input> element it will be processed by the framework. The input element will be broken into two pieces, one for **date** and one for **time**. If desired, the time portion can be suppressed using the attribute value 'time:suppress'.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date time: </td>
    <td><span oraField="dateTime" oraType="dateTime"></span></td>
  </tr>
  <tr>
    <td>Date only: </td>
    <td><span oraField="dateTime" oraType="dateTime; time:suppress"></span></td>
  </tr>
  <tr>
    <td>Date time: </td>
    <td><input oraField="dateTime" oraType="dateTime" /></td>
  </tr>
  <tr>
    <td>Date only: </td>
    <td><input oraField="dateTime" oraType="dateTime; time:suppress" /></td>
  </tr>
</table>
</body>
```

```

<xml>
<root>
<dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>

```

Is rendered as:

Date time: 11-01-2009 12:28AM
Date only: 11-01-2009
Date time:
Date only:

Figure 10: *oraType="dateTime" and oraType="dateTime; time:suppress" rendered as output and input*

oraType="dateTime; stdTime:true;"

The stdTime:true function is used to render a date / time element according to the daylight savings time schedule of the 'base' time zone. The 'base' time zone is specified on the Installation table and represents the database time zone.

NOTE: If you use oraType="dateTime; stdTime:true;" within an <input> element it will be processed by the framework. All time entered is assumed to correspond with the daylight savings time schedule of the base time zone. If a time is entered that cannot be unambiguously translated to standard time, then the user will be required to provide a time zone label to indicate whether daylight savings time, or standard time, has been entered.

```

<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date time: </td>
  <td><span oraField="dateTime" oraType="dateTime; stdTime:true;"></span></td>
  </tr>
  <tr>
    <td>Date time: </td>
  <td><input oraField="dateTime" oraType="dateTime; stdTime:true;" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>

```

Is rendered as:

NOTE: The time zone label is displayed because 1:28 is ambiguous otherwise. Legally, November 1, 2009 1:28 AM occurs twice because daylight savings time (DST) is removed at 2:00 AM. With the stdTime function time zone labels are only displayed when required to clarify time overlaps.

Date time: 11-01-2009 01:28AM PDT
Date time:

Figure 11: *oraType="dateTime; stdTime:true" rendered as output and input during DST overlap period*

A day later will be rendered as:

Date time: 11-02-2009 12:28AM

Date time:

Figure 12: `oraType="dateTime; stdTimeRef:true"` rendered as output and input for a day later than above

`oraType="dateTime; stdTimeRef:time zone xpath; displayRef:time zone xpath;"`

The `stdTimeRef:xpath` function is used to render a date / time element according to the daylight savings time schedule of a referenced time zone. Note that the time processed is assumed to have been stored in the standard time of the referenced time zone - so only daylight savings time shifting will execute - not time zone shifting.

The `displayRef:xpath` function is similar to the `stdTimeRef` function, except that `displayRef` will execute time zone shifting in addition to daylight savings time shifting. To use `displayRef` correctly, only associate it with time zone elements that have been stored in the base time zone.

NOTE: If you use `oraType="dateTime; stdTimeRef:xpath; displayRef:xpath"` within an `<input>` element it will be processed by the framework. All time entered is assumed to correspond with the daylight savings time schedule of the referenced time zone. If a time is entered that cannot be unambiguously translated to standard time, then the user will be required to provide a time zone label to indicate whether daylight savings time, or standard time, has been entered.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date time: </td>
    <td><span oraField="dateTime" oraType="dateTime; stdTimeRef:timeZone;"></span></td>
  </tr>
  <tr>
    <td>Date time: </td>
    <td><input oraField="dateTime" oraType="dateTime; stdTimeRef:timeZone;" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<timeZone>US-EAST</timeZone>
<dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>
```

Is rendered as:

NOTE: The time zone label is always displayed for a referenced time zone.

Date time: 11-01-2009 01:28AM EDT

Date time:

Figure 13: `oraType="dateTime; stdTimeRef:xpath"` rendered as output and input during DST overlap period

A day later will be rendered as:

Date time: 11-02-2009 12:28AM EST

Date time:

Figure 14: `oraType="dateTime; stdTimeRef:xpath"` rendered as output and input for a day later than above

`oraType="duration"`

The `oraType: duration` function is used to display time duration.

NOTE: If you use `oraType="duration"` within an `<input>` element it will be processed by the framework. For example, if you enter '90', it will be defaulted to '00:01:30' when you tab out of the input field.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Duration: </td>
    <td><span oraField="duration" oraType="duration"></span></td>
  </tr>
  <tr>
    <td>Duration: </td>
    <td><input oraField="duration" oraType="duration"/></td>
  </tr>
</table>
</body>
<xml>
<root>
<duration>90</duration>
</root>
</xml>
</html>
```

Is rendered as:

Duration: 00:01:30
Duration:

Figure 15: `oraType="duration"` rendered as output and input

`oraType="dayInMonth"`

The `oraType: dayInMonth` function is used to display a concatenated day and month.

NOTE: If you use `oraType="dateInMonth"` within an `<input>` element it will be processed by the framework.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Day In Month: </td>
    <td><span oraField="dayMonth" oraType="dayInMonth"></span></td>
  </tr>
  <tr>
    <td>Day In Month: </td>
    <td><input oraField="dayMonth" oraType="dayInMonth"/></td>
  </tr>
</table>
</body>
<xml>
<root>
<dayMonth>0228</dayMonth>
</root>
</xml>
</html>
```

Is rendered as:

Day In Month: 02-28
Day In Month:

Figure 16: `oraType="dayInMonth"` rendered as output and input

oraType="monthInYear"

The oraType: monthInYear function is used to display a concatenated month and year.

NOTE: If you use oraType="monthInYear" within an <input> element it will be processed by the framework.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Month In Year: </td>
    <td><span oraField="month" oraType="monthInYear"></span></td>
  </tr>
  <tr>
    <td>Month In Year: </td>
    <td><input oraField="month" oraType="monthInYear" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<month>200811</month>
</root>
</xml>
</html>
```

Is rendered as:

Month In Year: 11-2008

Month In Year:

Figure 17: oraType="monthInYear" rendered as output and input

oraType="money:CURRENCY | ;currencyRef:currency xpath"

The oraType: money function is used to display a number as a monetary amount. You can optionally specify a currency code directly (separated by a colon) or a reference to a currency code in the schema (separated by a semi-colon). If no currency or currency reference is specified then the installation currency will be used.

NOTE:

If you use oraType="money" within an <input> element it will be processed by the framework. For example, an error will be returned if a non-numeric value is entered. Additionally, you must specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header for right alignment:

- <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
- <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>

The stylesheet controls how the field will be rendered. If you want to alter the rendering you must override the oraMoney style.

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Amount, currency specified:</td>
    <td><span oraType="money:EUR" oraField="totalAmt"></span></td>
  </tr>
  <tr>
    <td>Amount, default currency:</td>
  </tr>
</table>
```

```

<td><span oraType="money" oraField="totalAmt"></span></td>
</tr>
<tr>
<td>Amount, default input:</td>
<td><input oraType="money" oraField="totalAmt" /></td>
</tr>
<tr>
<td>Amount, currency reference:</td>
<td><input oraType="money;currencyRef:cur" oraField="totalAmt" /></td>
</tr>
</table>
</body>
<xml>
<root>
<totalAmt>50500.09</totalAmt>
<cur>EUR</cur>
</root>
</xml>
</html>

```

Is rendered as:

Amount, currency specified:	€50,500.09
Amount, default currency:	\$50,500.09
Amount, default input:	<input type="text" value="\$50,500.09"/>
Amount, currency referenced:	€50,500.09

Figure 18: oraType="money" rendered

oraType="number"

The oraType: number function is used to display a number or validate an input value.

NOTE:

If you use oraType="number" within an <input> element it will be processed by the framework. For example, an error will be returned if a non-numeric value is entered. Additionally, you must specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header for right alignment:

- `<link rel="stylesheet" type="text/css" href="cisDisabled.css"/>`
- `<link rel="stylesheet" type="text/css" href="cisEnabled.css"/>`

The stylesheet controls how the field will be rendered. If you want to alter the rendering you must override the oraNumber style.

```

<html>
<head>
<link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
<link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
<tr>
<td>Count:</td>
<td><span oraType="number" oraField="count"></span></td>
</tr>
<tr>
<td>Count, input:</td>
<td><input oraType="number" oraField="count" /></td>
</tr>
</table>
</body>
<xml>
<root>
<count>989</count>
</root>
</xml>
</html>

```

Is rendered as:

Count:	989
Count, input:	<input type="text" value="989"/>

Figure 19: oraType="number" rendered

oraType="fkRef:true; info:true; context:true; navigation:true; search:true"

The oraType: fkRef function is used to enable functionality related to the foreign key reference meta-data, this includes: an information string, a context menu, and a hyperlink. Note that you can enable the foreign key hyperlink separately as well, see the [Embed Framework Navigation](#) section below for more information. The various attributes used to control foreign key reference functionality are as follows. Note that in every case, the default value is **true**. A value of **false** should be used to disable the feature.

- **fkRef** A value of 'true' enables all of the following foreign key reference processing. Use a value of 'false' to disable automatic foreign key reference processing.
- **info** A value of 'true' will render an information string on the UI map, if the foreign key meta-data is configured with an information program.
- **context** A value of 'true' will render a context menu to appear before the foreign key reference element, if the foreign key meta-data is configured for a menu.
- **navigation** A value of 'true' will cause the information string to be rendered as a URL, if the foreign key meta-data is configured with a navigation option.
- **search** A value of 'true' will cause a search zone to be rendered for input HTML elements, if the foreign key meta-data is configured with a search zone.

NOTE: Foreign key navigation and context menu functionality is only available for UI maps presented in a portal zone. UI Maps presented during BPA script processing cannot support navigation options. Search functionality is only available for input HTML elements.

- UI Map schema:

```
<schema>
  <userId fkRef="CI_USER" />
</schema>
```

- UI Map HTML:

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
<td>User</td>
<td><span oraField="userId" oraType="fkRef:true; info:true; context:true; navigation:true;"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
<userId>CZAAND</userId>
</root>
</xml>
</html>
```

- Is rendered as:

Figure 20: `oraType="fkRef:true; info:true; context:true; navigation:true"` rendered

`oraType="lookup:FIELD"`

The `oraType: lookup` function is used to display the description of a lookup value.

```

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Status:</td>
    <td><span oraField="status" oraType="lookup:batch_job_stat_flg"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <status>PD</status>
</root>
</xml>
</html>

```

Is rendered as:

Status:	Pending
---------	---------

Figure 21: `oraType="lookup:batch_job_stat_flg"` rendered

`oraType="lookupBO:BUS_OBJ_CD"`

The `oraType: lookupBO` function is used to display the description of a lookup business object value.

```

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Status:</td>
    <td><span oraField="status" oraType="lookupBO:status_bo"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <status>PD</status>
</root>
</xml>
</html>

```

Is rendered as:

Status:	Pending
---------	---------

Figure 22: `oraType="lookupBO:status_bo"` rendered

`oraType="charType:CHAR_TYPE_CD"`

The `oraType: charType` function is used to display the description of a predefined characteristic value.

```

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Skill:</td>

```



```

        <td><span oraType="charType:CM-SKILL" oraField="skill"></span></td>
      </tr>
    </table>
  </body>
</xml>
<root>
  <skill>10</skill>
</root>
</xml>
</html>

```

Is rendered as:

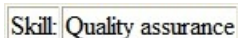


Figure 23: *oraType="charType:CM-SKILL" rendered*

oraType="table:CI_CURRENCY_CD"

The oraType: table function is used to display the description of a control table.

```

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
<td>Currency:</td>
<td><span oraType="table:CI_CURRENCY_CD" oraField="curr"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
<curr>USD</curr>
</root>
</xml>
</html>

```

Is rendered as:



Figure 24: *oraType="table:CI_CURRENCY_CD" rendered*

oraType="addGridRow, deleteGridRow"

The addGridRow function is used to build an "insert row" dialog into the UI map.

- An 'add' image will be displayed.
- Clicking on the image will insert a new row in the grid.
- If the list is empty, by default, an empty grid row will automatically be added. This means that the user will always see at least one grid row when this attribute is used.

The deleteGridRow function is used to build an "delete row" dialog into the UI map.

- A 'delete' image will be displayed.
- Clicking on the image will remove the adjacent row from the grid.

NOTE: Because add and delete dialogs are relevant only inside a table, these attributes must be specified within a <td> element.

CAUTION: The addGridRow and deleteGridRow attributes are designed to work with the business object action of 'replace' rather than 'update'. Therefore, if the map contains a grid, the business object action of 'replace' must be used to update the business object. Refer to [BO Replace Action](#) for more information.

Source:

```
<html>
<head>
<title>Demonstrate Grid Add and Grid Delete OraTypes</title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table oraList="listEntry">
  <thead>
    <tr>
      <th/>
      <th/>
    <th><span>Date</span></th>
    <th><span>Amount</span></th>
  </thead>
  <tr>
    <td oraType="addGridRow"></td>
    <td oraType="deleteGridRow"></td>
    <td>
      <input oraField="date" oraType="date"></input>
    </td>
    <td align="right">
      <input oraField="amount" oraType="money"></input>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
  <listEntry>
<date>2008-01-01</date>
<amount>44.28</amount>
  </listEntry>
  <listEntry>
<date>2008-02-01</date>
<amount>32.87</amount>
  </listEntry>
  <listEntry>
<date>2008-03-01</date>
<amount>21.76</amount>
  </listEntry>
</root>
</xml>
</html>
```

Is rendered as:

	Date		Amount
 	01-01-2008	 	\$44.28
 	02-01-2008	 	\$32.87
 	03-01-2008	 	\$21.76

Figure 25: oraType="addGridRow" and "deleteGridRow" rendered

oraType="raw"

The oraType: raw function is used to display the contents of an element that contains 'raw' data as defined for the schema element being rendered.

- UI Map schema:

```
<schema>
  <rawStuff type="raw"/>
</schema>
```

- UI Map HTML:

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Raw Stuff:</td>
    <td><span oraType="raw" oraField="rawStuff"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
  <rawStuff><ele1>text in element 1</ele1><group1><ele2>text inside element 2, group 1</
ele2><ele3>text inside element 3, group 1</ele3></group1></rawStuff>
</root>
</xml>
</html>
```

Is rendered as:

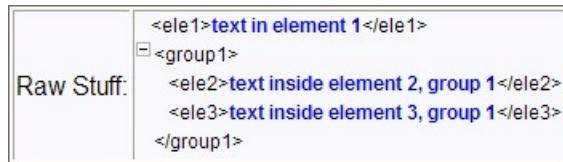


Figure 26: oraType="raw" rendered

oraType="xmlString"

The oraType: xmlString function is used to display the contents of an element, as XML pretty-print, when the element contains escaped XML.

NOTE:

It is not required, but the pretty print of the rendered XML is enhanced if you specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header:

- <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
- <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>XML Stuff:</td>
    <td><span oraType="xmlString" oraField="xmlStuff"></span></td>
  </tr>
</table>
</body>
```

```

<xml>
<root>
<xmlStuff><ele1>text in element 1</ele1><group1><ele2>text inside element 2, group 1</
ele2><ele3>text inside element 3, group 1</ele3></group1></xmlStuff>
</root>
</xml>
</html>

```

Is rendered as:

XML Stuff:	<pre> <ele1>text in element 1</ele1> <group1> <ele2>text inside element 2, group 1</ele2> <ele3>text inside element 3, group 1</ele3> </group1> </pre>
------------	--

Figure 27: oraType="xmlString" rendered

Without rendering:

XML Stuff:	<pre> <ele1>text in element 1</ele1><group1><ele2>text inside element 2, group 1</ele2><ele3>text inside element 3, group 1</ele3></group1> </pre>
------------	--

Figure 28: XML contents rendered without oraType="xmlString"

oraType="html"

The oraType: html function is used to display the contents of an element as HTML as opposed to plain text. An element defined as oraType="fkref" is automatically rendered as HTML.

WARNING:

To avoid execution of malicious HTML not all HTML tags are supported. The list of supported tags is defined in the "F1-HTMLWhiteList" managed content definition.

If unsupported HTML is detected the entire element is escaped and rendered as plain text. It is therefore recommended to properly escape any source string that contributes to the final HTML element if it is not expected to contain valid HTML. This way only the offending string is escaped and not the entire element.

If the HTML element is composed in scripting refer to the 'escape' function described in the [Edit Data Syntax](#) for more information. Use the WebStringUtilities.asHTML java API for escaping text composed in Java.

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Info :</td>
    <td><span oraType="html" oraField="info"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
  <info><b>text in bold</b></info>
</root>
</xml>
</html>

```

Is rendered as:

Info : **text in bold**

Figure 29: oraType="html" rendered

Without rendering:

Info : text in bold

Figure 30: HTML contents rendered without oraType="html"

Search Using a Pop-Up Explorer Zone

oraSearch="Search Zone"

The oraSearch attribute is used to enable search zone functionality for input HTML elements.

- 'Search Zone': The name of the explorer zone to use for the search.

NOTE: The oraSearch attribute is similar to the oraType attribute, because it will be 'automatically' included into HTML via the oraSchemaDataTypes attribute. This means that coding the oraSearch attribute into UI Map HTML is only required if a search zone has not been specified in the schema, or in the schema element's fkRef.

- Source with oraSearch declared in the HTML:

```
<schema>
  <uiMap/>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap" oraSearch="F1-UISRCH"></td>
  </tr>
</table>
</body>
</html>

<xml>
<root>
  <uiMap/>
</root>
</xml>
</html>
```

Is rendered as:

UI Map with Search

Figure 31: oraSearch rendered as a search button

- Source with a search declared in the schema:

```
<schema>
  <uiMap search="F1-UISRCH"/>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap"></td>
  </tr>
</table>
</body>
</html>
```

```

<root>
  <uiMap/>
</root>
</xml>
</html>

```

Is rendered as:



Figure 32: Schema search rendered as a search button

- Source with a search declared in the schema element's fkRef:

```

<schema>
  <uiMap fkRef="F1-UISRC" />
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap"></td>
  </tr>
</table>
</body>
<xml>
<root>
  <uiMap/>
</root>
</xml>
</html>

```

Is rendered as:

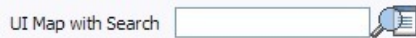


Figure 33: fkRef search zone rendered as a search button

oraSearchField="field name:field value; ..."

The oraSearchField attribute is used to initialize search zone filters - it is optional. Multiple filters may be initialized. This attribute can only be used in conjunction with the oraSearch attribute.

1. *'field name'*: The field name is used to identify the zone filter to initialize when the search is launched. The field name must match the value of the searchField mnemonic specified on a search zone filter parameter.
2. *'field value'*: The field value is used to determine the value used to initialize the filter with. A field value can be omitted, or have one of two different formats. Possible values:
 - None. If you do not specify a field value, then the value of the input element containing the oraSearchField attribute will be used.
 - xpath. You can specify xpath to the UI Map schema element that contains the value to use.
 - literal. You can specify a literal value, in quotes.

NOTE: The oraSearchField attribute is optional when oraSearch is used. If you do not specify an oraSearchField attribute, and the schema element has a search enabled fkRef specified, the framework will automatically build an oraSearchField where the field name is equal to the fkRef's key field.

WARNING: The pop-up explorer zone can be invoked one of two ways: By clicking on the search button, or by hitting the enter key from the field to the left of the button. If you click on the button then no search field information will be passed to the zone. Search field information is only used to initialize zone filter values when enter is pressed.

Two filter values are initialized as shown in the following example:

```
<schema>
  <bo/>
  <uiMap/>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap" oraSearch="F1-
UISRCH" oraSearchField="MAP_CD; BUS_OBJ_CD:bo;"></td>
  </tr>
</table>
</body>
</html>

<xml>
<root>
  <bo/>
  <uiMap/>
</root>
</xml>
</html>
```

oraSearchOut="field name:xpath target; ..."

The oraSearchOut attribute is used to direct values returned by the search zone - it is optional. Multiple oraSearchOut fields may be specified. This attribute can only be used in conjunction with the oraSearch attribute.

1. *'field name'*: The field name is used to identify the search result returned from the query zone. The field name must match the ELEMENT_NAME mnemonic defined within the explorer zone's search results parameter.
2. *'xpath target'*: The xpath target is used to identify the UI Map schema element that will receive the value returned by the search zone. Possible values:
 - a. None. If you do not specify a target, then the input element containing the oraSearchOut attribute will be used.
 - b. xpath. You can specify xpath to the UI Map schema element to receive the returned value.

NOTE: The oraSearchOut attribute is optional when oraSearch is used. If you do not specify an oraSearchOut attribute, the framework will build a default where the field name will be set equal to the oraSearchField's field name.

Two values are returned in the following example:

```
<schema>
  <bo/>
  <mo/>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>BO Search </td>
    <td><input oraField="bo" oraSearch="Z1-BOSRCH" oraSearchOut="BUS_OBJ_CD; MO_CD:mo;"></td>
  </tr>
</table>
</body>
</html>

<xml>
<root>
  <bo/>
  <mo/>
</root>
```

```
</xml>
</html>
```

Enable UI Map Help

oraHelp="field:FIELD"

The oraHelp attribute is used to configure tool tip help text. A tool tip help icon will be displayed, which the user can click to read help text. The help text must be defined on a meta-data field used as a label in the map.

By default, the oraHelp attribute is automatically applied to all elements in the map that are linked to an [oraLabel](#) or [oraMdLabel](#) attribute where the underlying field contains help text. To avoid automatic oraHelp processing for a particular element, enter: oraHelp="".

- field: Identifies the field that contains the help text displayed for the tool tip.

WARNING: To automatically enable the oraHelp icon on the map, either the oraLabel or the oraMdLabel attribute must be used.

NOTE:

Help text must be entered on the field. The field can be linked to the map's schema in two different ways: by the mapField attribute or via the mdField attribute (refer to [Schema Nodes and Attributes](#) for more information).

It is possible to change the oraHelp automatic processing. Refer to [Custom Look And Feel Options](#) for more information

- The following is a UI Map schema in which help text is automatically rendered for the UI map element that specifies the address element using the oraLabel attribute. Note that the following example would render identically if the mapField attribute were specified in the schema instead of mdField.

```
<schema>
  <address mdField="ADDRESS" />
  <city/>
  <state/>
  <zip/>
</schema>
```

- HTML source for oraHelp tooltip automatically applied using the schema defined above for the address element. Also, automatic tooltip help applied via oraMdLabel for the field 'ADDRESSHEADER'.

```
<html>
<body>
<table>
  <tr>
    <td colspan="2" oraMdLabel="ADDRESSHEADER"></td>
  </tr>
  <tr/>
  <tr/>
  <tr>
    <td oraLabel="address"></td>
    <td><input type="text" oraField="address"/></td>
  </tr>
  <tr>
    <td>City</td>
    <td><input type="text" oraField="city"/></td>
  </tr>
  <tr>
    <td>State</td>
    <td><input type="text" oraField="state"/></td>
  </tr>
  <tr>
    <td>Zip</td>
    <td><input type="text" oraField="zip"/></td>
  </tr>
</table>
```



```

</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

- HTML source for oraHelp tooltip **manually** applied. This example is for demonstration purposes only - it should rarely be employed. The ADDRESS meta-data field contains the help text: 'Enter street number...'.

```

...
  <tr>
    <td>Address</img></td>
    <td><input type="text" oraField="address"/></td>
  </tr>
...

```

HTML rendered:

Enter Address Information ?

Address ?

City

State

Zip

Figure 34: oraHelp tool tip rendered for meta-data field help text

Display Labels

oraLabel="element xpath"

The oraLabel attribute is used to obtain a language sensitive label for a , <td>, or <input> HTML element. The label text is derived from the UI map's schema, where the schema element must have one of the following schema element attributes specified: mapField, mdField, or label.

NOTE: You can also define a field directly in your HTML for label definition, refer to [oraMdLabel](#) for more information.

WARNING: In addition to the HTML attribute, the oraLabel attribute also requires a UI Map schema attribute of mapField="Column Name", mdField="Column Name", or label="Label Text". You must have at least one of these elements present in order to use the oraLabel HTML attribute.

NOTE: If the schema contains multiple attributes, the oraLabel attribute will pick the label to render according to the following hierarchy: The label attribute overrides the mdField attribute, which in turn will override the mapField attribute.

- UI Map schema:

```

<schema>
  <user mapField="USER_ID" />
  <info type="group" mapXML="CLOB">
    <city label="Metro Area" />
    <age mdField="AGE_LBL" />
  </info>
</schema>

```

- HTML:

```

<html>
<head><title oraMdLabel="BUS_LBL"></title></head>
<body>
<table>
  <tr>
    <td oraLabel="user"></td>
    <td><input oraField="user" /></td>
  </tr>
  <tr>
    <td oraLabel="info/city"></td>
    <td><input oraField="info/city" /></td>
  </tr>
  <tr>
    <td oraLabel="info/age"></td>
    <td><input oraField="info/age" /></td>
  </tr>
  <tr>
    <td />
    <td><input type="button" oraMdLabel="ACCEPT_LBL" /></td>
  </tr>
</table>
</body>
<xml>
  <root>
    <user>RWINSTON</user>
    <info>
      <city>Alameda</city>
      <age>32</age>
    </info>
  </root>
</xml>
</html>

```

HTML rendered:

User	<input type="text" value="RWINSTON"/>
Metro Area	<input type="text" value="Alameda"/>
Age	<input type="text" value="32"/>
	<input type="button" value="Accept"/>

Figure 35: oraLabel elements rendered

oraMdLabel="FIELD"

The oraMdLabel attribute is used to obtain a language sensitive label for a , <td>, <input>, or <title> HTML element. The label text is derived from the field referenced.

NOTE: You can also define labels derived from the map's schema definition, refer to [oraLabel](#) for more information.

You can use the oraMdLabel attribute to create a language independent title for a pop-up UI map. To do this, you simply add the oraMdLabel attribute to the <title> attribute.

- HTML:

```

<html>
<head><title oraMdLabel="NOTES_LBL"></title></head>
<body>
<table>
  <tr>
    <td oraLabel="user"></td>
    <td><input oraField="user" /></td>
  </tr>
  <tr>

```

```

        <td oraLabel="info/city"></td>
        <td><input oraField="info/city"/></td>
    </tr>
    <tr>
        <td oraLabel="info/age"></td>
        <td><input oraField="info/age"/></td>
    </tr>
    <tr>
        <td/>
        <td><input type="button" oraMdLabel="ACCEPT_LBL"/></td>
    </tr>
</table>
</body>
<xml>
    <root>
        <user>RWINSTON</user>
        <info>
            <city>Alameda</city>
            <age>32</age>
        </info>
    </root>
</xml>
</html>

```

HTML rendered:

Display Errors

oraErrorVar="ERRMSG-TEXT"

The oraErrorVar attribute is used to display an error variable. You can choose to display any of the following error variables:

- ERRMSG-TEXT
- ERRMSG-LONG
- ERRMSG-CATEGORY
- ERRMSG-NUMBER

```

...
<table width="100%" cellpadding="12">
    <tr class="oraErrorText">
        <td>
            <a href="" onclick="oraShowErrorAlert(); return false;">
<span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
            </a>
        </td>
    </tr>
</table>
...

```

Is rendered as:

Last name required

User Id BOND007

First Name

Last Name

Figure 36: oraErrorVar rendered

oraError="automate:true; prefix:xpath;"

The oraError attribute is used within the body tag of an input UI map to automate oraErrorElement processing for all input elements of the map (see below for more information on oraErrorElement processing).

- automate: Specify 'true' to automate oraErrorElement processing. Use 'false' to turn off automatic processing. Default is 'true' - which means that by default, the map will have automatic oraErrorElement processing built into it.
- prefix: Automatic oraErrorElement processing will cause an oraErrorElement attribute to be injected into every UI map element that has an existing oraField attribute (but no existing oraErrorElement attribute). By default, the oraErrorElement attribute value is set exactly equal to the oraField attribute value. However, if the error element values thrown by the framework are missing a prefix that the map's oraField attributes contain, you should specify it here (see example below). The prefix specified will be removed from the oraField attribute value before creation of the automatic oraErrorElement attribute value.

NOTE: By default, oraError="automate:true" will be injected into UI maps during rendering. To disable automatic processing, you must specify the following in the body tag of the UI map: <body oraError="automate:false;"/>

NOTE:

A pair of stylesheet references, cisEnabled and cisDisabled, must be specified for reference of the oraError style:

- <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
- <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>

The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the oraError style.

HTML:

```
<html>
<head>
  <title>User Zone Input</title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body oraError="automate:true; prefix:boGroup">
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td>
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
      </a>
    </td>
  </tr>
</table>
<table width="100%" border="0" cellpadding="4">
  <tr style="padding-top:30px;">
    <td align="right" class="label">User Id</td>
    <td>
      <span oraField="boGroup/userId" class="normal"/>
    </td>
  </tr>
</table>
```

```

<td align="right" class="label">First Name</td>
<td>
  <input oraField="boGroup/firstName" class="normal"/>
</td>
</tr>

<tr>
<td align="right" class="label">Last Name</td>
<td>
  <input oraField="boGroup/lastName" class="normal"/>
</td>
</tr>
</table>
</body>

<xml>
<root>
<boGroup>
  <userId>BOND007</userId>
  <firstName>James</firstName>
  <lastName></lastName>
</boGroup>
</root>
</xml>
</html>

```

Is rendered as shown, where the error element thrown is equal to 'lastName':

Figure 37: oraErrorElement automatically rendered using oraError in the body tag

oraErrorElement="element name"

The oraErrorElement attribute is used in conjunction with the cisDisabled stylesheet to highlight a field in error.

- *element name*: The element name referenced by the oraErrorElement must exactly match the name of the error element assigned when the error was thrown. More than one HTML field can be referenced by the same error element name.

NOTE:

For more information on throwing an error, refer to the Terminate statement in the [Edit Data Syntax](#).

Example of service script step that defines an error element:

```
terminate with error (90000, 207 %1='Last name required' element='lastName');
```

NOTE:

A pair of stylesheet references, cisEnabled and cisDisabled, must be specified for reference of the oraError style:

```
<link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
```

```
<link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
```

The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the oraError style.

HTML:

```

<html>
<head>
  <title>User Zone Input</title>

```

```

<link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
<link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td>
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
      </a>
    </td>
  </tr>
</table>
<table width="100%" border="0" cellpadding="4">
  <tr style="padding-top:30px;">
    <td align="right" class="label">User Id</td>
    <td>
      <input oraField="userId" class="normal" />
    </td>
  </tr>

  <tr>
    <td align="right" class="label">First Name</td>
    <td>
      <input oraField="firstName" class="normal" oraErrorElement="firstName" />
    </td>
  </tr>

  <tr>
    <td align="right" class="label">Last Name</td>
    <td>
      <input oraField="lastName" class="normal" oraErrorElement="lastName" />
    </td>
  </tr>
</table>
</body>

<xml>
<root>
  <userId>BOND007</userId>
  <firstName>James</firstName>
  <lastName></lastName>
</root>
</xml>
</html>

```

Is rendered as:

The screenshot shows a form with three fields. At the top, there is a purple error message: "Last name required". Below it, the "User Id" field contains the value "BOND007". The "First Name" field contains the value "James". The "Last Name" field is currently empty and has a red background, indicating it is required and has not been filled out.

Figure 38: oraErrorElement rendered

oraShowErrorAlert(); return false;

The oraShowErrorAlert function is used to pop-up the standard error dialog. See below for an example.

In the example below the error will be displayed as a URL, and clicking the URL will cause the standard error pop-up to be displayed.

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>

```

```

<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td>
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
      </a>
    </td>
  </tr>
</table>
<table>
  <tr>
    <td>Address: </td>
    <td><input type="text" oraField="address"/></td>
  </tr>
  <tr>
    <td>City: </td>
    <td><input type="text" oraField="city"/></td>
  </tr>
  <tr>
    <td>State: </td>
    <td><input type="text" oraField="state"/></td>
  </tr>
  <tr>
    <td>Zip: </td>
    <td><input type="text" oraField="zip"/></td>
  </tr>
  <tr>
    <td/>
    <td style="padding-top:15px;">
<oraInclude map="F1-SaveCancelButton"/>
    </td>
  </tr>
</table>
</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

Error text displayed via oraErrorVar:

Address field missing

Address:

City:

State:

Zip:

Figure 39: Error message rendered using oraErrorVar

Standard error pop-up dialog launched via click on error message:

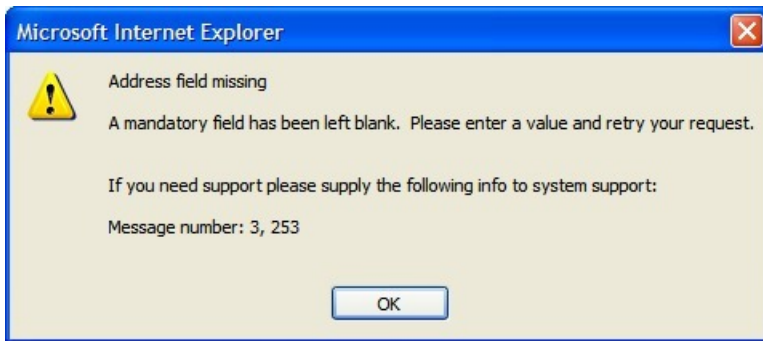


Figure 40: Detailed error message launched using oraShowAlert function

Fire JavaScript for Browser Events

Working with the JavaScript Framework

There are many javascript events that can be used within the HTML/Javascript environment. These include events such as onLoad, onBlur, onChange, etc. The UI Map Framework also makes use of some of these events. It is important that any UI Map you develop works with the framework in order to obtain consistent results (events may not always be executed in the same order at all times!).

WARNING:

The following describes the recommended approach for safely handling loading and processing field updates in your UI Maps.

If javascript is required within an XHTML UI Map or fragment, it will be necessary to bound it within a `![CDATA[]]` tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
```

```
/* <![CDATA[ */
```

```
//
```

```
// javascript
```

```
//
```

```
/* ]]> */
```

```
</script>
```

oraChange="function"

The oraChange HTML attribute provides a method of executing a JavaScript function during element change.

At the time of UI Map load, if there is an event handler(s) already attached to an HTML element, the framework removes it and attaches a combined event handler. The combined handler calls any **framework handler first** and then calls the other (custom) handlers.

WARNING: Note that the function must not be used to execute logic that will modify the associated field data value again - or an endless loop will occur!

In the following example the oraInvokeBS function is executed when the button is clicked.

```
<html>
  <head>
```



```

        <title>oraInvokeBS test</title>
    </head>
    <body>
        <table>
            <tr>
                <td>User Id:</td>
                <td>
                    <input oraField= "xmlGroup/userId" />
                    <input type="button" value="Search" oraChange="oraInvokeBS('UserSearch', 'xmlGroup');" />
                </td>
            </tr>
            <tr>
                <td/>
                <td>Search Results</td>
            </tr>
            <tr>
                <td/>
                <td>
                    <table oraList="xmlGroup/searchList">
<tr><td><span oraField="userId"></span></td></tr>
                    </table>
                </td>
            </tr>
        </table>
    </body>
    <xml>
        <root>
            <xmlGroup>
                <userId/>
                <searchList>
                    <userId></userId>
                </searchList>
            </xmlGroup>
        </root>
    </xml>
</html>

```

oraLoad="function"

The oraLoad HTML attribute provides a method of executing a JavaScript function during page load.

WARNING: When executing oraLoad within a fragment UI map, and you need to execute a JavaScript function during page load (where the function invokes a business object, business service, or service script) you can use the special syntax "oraLoad[\$SEQUENCEID]". For other special syntax used for map fragments, refer to the [Construct a Portal Zone Header](#) section.

- In the following example, the [oraDisplayNone](#) function is executed during page load:

```

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
        <td oraLoad="oraDisplayNone(item, 'userId', '')">User Id: </td>
        <td><span oraField="userId"></span></td>
    </tr>
</table>
</body>
<xml>
<root>
    <userId>SPLAXT</userId>
</root>
</xml>
</html>

```

- Is rendered as:

User Id: SPLAXT

Figure 41: oraLoad rendered

oraAfterLoad="function"

The oraAfterLoad HTML attribute provides a method of executing a JavaScript function after oraLoad.

In the following example the oraGetValueFromScript function is executed during page load.

```
<div>
  <label for="boGroup_parameters_externalSystem" oraLabel="boGroup/parameters/
externalSystem"></label>
  <span>
    <select oraSelect="bs:F1-
RetrieveExternalSystems" class="oraInput" id="boGroup_parameters_externalSystem" oraField="boGroup/
parameters/externalSystem" oraSelectOut="valuePath:results/externalSystem; descPath:results/
description" oraSelectIn="outboundMsgType:boGroup/parameters/
outboundMsgType" oraAfterLoad="oraGetValueFromScript(document.getElementById('boGroup_parameters_externalSyst
select>
    </span>
  </div>
```

Hide Elements

oraDisplayNone(item,'xpath' | function, 'value','oper');

The oraDisplayNone function is used to hide an HTML element based upon the value of an XML element or the result of a JavaScript function.

1. *item*: the first parameter must always contain the string 'item' (it lets the JavaScript function identify the HTML item being manipulated).
 2. *'xpath'*: Identifies an element via XPath.
 3. *function*: The name of a JavaScript function. This function must return a Boolean, which means the 'value' used with a JavaScript function must be true or false.
 4. *'value'*: The value to compare against the value in the XPath.
 5. *'operator'*: You can optionally supply an operator in to evaluate the element against the value. For example, '!=', '>', or '<'. By default, the operator is '='.
- In the following example the User Id label will be hidden when no user Id value exists (when the userId = '')

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td oraLoad="oraDisplayNone(item,'userId','')">User Id: </td>
    <td><span oraField="userId"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
  <userId></userId>
</root>
</xml>
</html>
```

- In the following example the Save button will be hidden when the user does not have security access to the action of change for the application service of 'F1-DFLTS'.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td oraLoad="oraDisplayNone(item, oraHasSecurity('F1-DFLTS', 'C'), false );">
      <input name="Save" type="button" onclick="oraInvokeBO('CM-
IndividualTaxpayer', null, 'replace')"/>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
  <userId></userId>
</root>
</xml>
</html>
```

Is rendered as:



Figure 42: *oraDisplayNone rendered (nothing to show!)*

oraHasSecurity('Application Service','Access Mode');

The oraHasSecurity JavaScript function returns a Boolean value. If the logged on user has security access to the application service's access mode then true is returned, otherwise false is returned.

1. *'Application Service'*: The prime key of an application service.
2. *'Access Mode'*: A valid access mode of the application service.

In the following example the Save button will be hidden when the user does not have security access to the action of change for the application service of 'F1-DFLTS'.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td oraLoad="oraDisplayNone(item, oraHasSecurity('F1-DFLTS', 'C'), false );">
      <input name="Save" type="button" onclick="oraInvokeBO('CM-
IndividualTaxpayer', null, 'replace')"/>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
<userId></userId>
</root>
</xml>
</html>
```

oraHasSecurityPath('Application Service xpath','Access Mode xpath');

The oraHasSecurityPath JavaScript function returns a Boolean value. If the logged on user has security access to the application service's access mode then true is returned, otherwise false is returned.

1. *'Application Service xpath'*: Defines the xpath location of an application service.
2. *'Access Mode xpath'*: Defines an xpath location of an access mode of the application service.

In the following example the button will be hidden when the user does not have security access to the access mode 'ACT' of the application service 'FORMTST'.

```
<html>
<body>
<table>
  <tr>
    <td oraLoad="oraDisplayNone(item, oraHasSecurityPath('appService', 'accessMode'), false );">
      <input oraField="statusLabel" type="button" onclick="oraRunScript('UpdateState','status');"/>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
  <status>ACTIVATE</status>
<statusLabel>Activate</statusLabel>
<appService>FORMTST</appService>
  <accessMode>ACT</accessMode>
</root>
</xml>
</html>
```

Invoke Schema Based Services

oraInvokeBO('BO name','xpath' | null,'action');

The oraInvokeBO function is used to perform a BO interaction directly from the UI map's HTML.

1. *'BO name'*: the name of the business object to invoke.
2. *'xpath'*: Identifies a group element via XPath. If you specify the word **null**, then the entire embedded XML object will be passed.
3. *'action'*. Either: 'add', 'delete', 'read', 'update', 'replace'

NOTE: The oraInvokeBO statement returns a 'true' or a 'false' depending on whether the invocation encounters an error.

- An example of the oraInvokeBO statement coded within a JavaScript function.

```
function invokeBO {
  if (!oraInvokeBO('User', 'xmlGroup', 'read')) {
    oraShowErrorAlert();
    return;
  }
}
```

- In the following example the business object 'User' will be invoked with the group element 'xmlGroup' and action 'read'.

```
<html>
<head>
  <title>oraInvokeBO test</title>
</head>
<body>
  <table>
    <tr>
      <td>User Id:</td>
      <td>
        <input oraField= "xmlGroup/userId"/>
        <input type="button" value="Find" onClick="oraInvokeBO('User', 'xmlGroup', 'read');"/>
      </td>
    </tr>
    <tr>
      <td/>
      <td>Result</td>
    </tr>
  </table>
</body>
</html>
```

```

        <td/>
        <td>
            <span oraField="xmlGroup/firstName"></span>
            <span oraField="xmlGroup/lastName"></span>
        </td>
    </tr>
</table>
</body>

<xml>
  <root>
    <xmlGroup>
      <userId>SPLNXU</userId>
      <firstName></firstName>
      <lastName></lastName>
    </xmlGroup>
  </root>
</xml>
</html>

```

Is rendered as:

User Id:

Result

Czarina Andrada

Figure 43: *oralInvokeBO rendered*

oralInvokeBS('BS name','xpath' | null);

The oraInvokeBS function is used to perform a business service interaction directly from the UI map's HTML.

1. *'BS name'*: the name of the business service to invoke.
2. *'xpath'*: Identifies a group element via XPath. If you specify the word **null**, then the entire embedded XML object will be passed.

NOTE: The oraInvokeBS statement returns a 'true' or a 'false' depending on whether the invocation encounters an error.

- An example of the oraInvokeBS statement coded within a JavaScript function.

```

function invokeBS {
  if (!oraInvokeBS('UserSearch', 'xmlGroup')) {
    oraShowErrorAlert();
    return;
  }
}

```

- In the following example the business service 'UserSearch' will be invoked with the group element 'xmlGroup'.

```

<html>
  <head>
    <title>oraInvokeBS test</title>
  </head>
  <body>
    <table>
      <tr>
        <td>User Id:</td>
        <td>
          <input oraField= "xmlGroup/userId"/>
          <input type="button" value="Search" onClick="oraInvokeBS('UserSearch', 'xmlGroup') ;"/>
        </td>
      </tr>
      <tr>
        <td/>
        <td>Search Results</td>
      </tr>
    </table>
  </body>
</html>

```

```

        </tr>
        <tr>
            <td/>
            <td>
                <table oraList="xmlGroup/searchList">
<tr><td><span oraField="userId"></span></td></tr>
                </table>
            </td>
        </tr>
    </table>
</body>
</xml>
<root>
    <xmlGroup>
        <userId/>
        <searchList>
            <userId></userId>
        </searchList>
    </xmlGroup>
</root>
</xml>
</html>

```

Is rendered as:

User Id:

Search Results

BBLABY
 CBALKAN
 CRUPPERT
 CTICZON
 DSISKA

Figure 44: oraInvokeBS rendered

oraInvokeSS('service script name','xpath' | null);

The oraInvokeSS function is used to perform a service script interaction directly from the UI map's HTML.

- 'SS name': the name of the service script to invoke.
- 'xpath': Identifies a group element via XPath. If you specify the word **null**, then the document belonging to the parent node will be passed. If the parent node is not enough, then the entire document can always be passed using the following syntax:

```
oraInvokeSS('service script', null, null, [$SEQUENCEID])
```

NOTE: The oraInvokeSS statement returns a 'true' or a 'false' depending on whether the invocation encounters an error.

- An example of the oraInvokeSS statement coded within a JavaScript function:

```
function invokeSS {
    if (!oraInvokeSS('UserSearch', 'xmlGroup')) {
        oraShowErrorAlert();
        return;
    }
}
```

- In this example, the service script 'UserSearch' is invoked with the group element 'xmlGroup'.

```
<html>
  <head>
    <title>oraInvokeSS test</title>
  </head>
  <body>
```

```

        <table>
          <tr>
            <td>User Id:</td>
            <td>
              <input oraField= "xmlGroup/userId" />
              <input type="button" value="Search" onClick="oraInvokeSS('UserSearch','xmlGroup');"/>
            </td>
          </tr>
          <tr>
            <td/>
            <td>Search Results</td>
          </tr>
          <tr>
            <td/>
            <td>
              <table oraList="xmlGroup/searchList">
                <tr><td><span oraField="userId"></span></td></tr>
              </table>
            </td>
          </tr>
        </table>
      </body>
    </xml>
  </root>
</xml>
</html>

```

The code is rendered as:

User Id:

Search Results

BBLABY
 CBALKAN
 CRUPPERT
 CTICZON
 DSISKA

Figure 45: oraInvokeSS rendered

Refresh a Rendered Map or Portal Page

oraRefreshMap;

The oraRefreshMap function is used to execute a 'Refresh' command from a UI map rendered in a zone. Only the map zone issuing the command will be refreshed.

```

...
  <tr>
    <td/>
    <td><input type="button" onclick="oraRefreshMap();" value="Refresh"/></td>
  </tr>
...

```

oraRefreshPage;

The oraRefreshPage function is used to execute the 'Refresh Page' command from a UI map rendered in a zone. All zones of the portal will be refreshed.

```
...
  <tr>
    <td/>
    <td><input type="button" onclick="oraRefreshPage();" value="Refresh"/></td>
  </tr>
...
```

Embed Framework Navigation

oraNavigate('Navigation Option','key xpath');

The oraNavigate function is used to execute a page transfer using the navigation information defined on a navigation option.

CAUTION: The oraNavigate function is only intended for a UI map defined within a portal zone, and it should not be used within a UI map launched by a BPA script.

The following example exhibits two possible uses of the oraNavigate function: as a URL and as a button. Note that the UI Map schema must contain a fkRef attribute for the fkRef rendering - refer to the oraType="fkRef" HTML attribute for more information.

```
<schema>
  <userId fkRef="CI_USER" />
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>User Link: </td>
    <td><a href="" onclick="oraNavigate('userMaint','userId'); return false;"><span oraField="userId" ora
span></a></td>
  </tr>
  <tr>
    <td>User Button: </td>
    <td><input type="submit" onclick="oraNavigate('userMaint','userId')" value="Go to User"/
></td>
  </tr>
</table>
</body>
<xml>
<root>
  <userId>SPLAXT</userId>
</root>
</xml>
</html>
```

The code is rendered as:

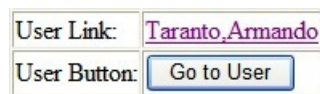


Figure 46: oraNavigate rendered as URL and button

Launch BPA Script

oraRunScript('BPA Script','xpath','xpath');

The oraRunScript function is used to launch a BPA script. In addition to launching the script, you can push XML element values into temporary storage. The BPA script can then reference the temporary variables by element name.

CAUTION: The oraRunScript function is only applicable to UI maps displayed in portal zones. UI maps launched within a running BPA script cannot directly launch another BPA script from the UI map's HTML. Instead, return a value from the UI map and execute a Perform Script or Transfer Control step type.

NOTE: It is incumbent on the script author to pull information out of temporary storage in the initial steps of the script.

In the following example, a temporary variable named 'personId' is created with value '1234567890' and the BPA script named 'Edit Address' is launched.

```
<html>
<body>
<table>
  <tr>
    <td>
      <div oraField="address"></div>
      <span oraField="city"></span>
      <span>,</span>
      <span oraField="state"></span>
      <span oraField="zip"></span>
      <span oraField="country"></span>
      <a href="" onClick="oraRunScript('Edit Address','personId');">edit</a>
    </td>
  </tr>
</table>
</body>
<xml>
  <root>
    <personId>1234567890</personId>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>
```

HTML for BPA script href is rendered as follows:

123 Main St
Alameda, CA 94770 [edit](#)

Figure 47: href for BPA Script

oraRunScriptWithValues('BPA Script','name':value);

The oraRunScriptWithValues function is used to launch a BPA - typically from a custom JavaScript function. In addition to launching the script, you can push name/value pairs into temporary storage. Multiple values can be passed. The BPA script can then reference the temporary variables by name.

NOTE: You would use this JavaScript function, instead of oraRunScript, if you need to push values to the BPA script that are not located in the UI Map's XML structure.

In the example below, a JavaScript function named 'editUser()' is responsible for launching the BPA script named 'UserEdit'. The temporary variable named 'userId' will be created with value 'CMURRAY'.

```
<html>
<head>
<script type="text/javascript">

function editUser() {
  var values = {'userId': 'CMURRAY'};
  oraRunScriptWithValues('UserEdit', values);
  return;
}
```

```

</script>
</head>
<body>
...
</body>
</html>

```

Exit UI Map with Bound Values

oraSubmitMap('value', false);

The `oraSubmitMap` function is used to exit a UI Map. When you quit the map you can specify a value to return to the script and, in addition, whether to return updated XML. For example, if you want to quit the map in "Cancel" mode, then you should specify the additional argument of *false*. By default, XML will be returned (default is *true*).

In the following example, the Save button will return updated information, the Cancel button will not.

```

<html>
<body>
<table>
  <tr>
    <td/>
    <td style="padding-bottom:15px;">
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span oraErrorVar="ERRMSG-TEXT"></span></a>
      </td>
    </tr>
    <tr>
      <td >Address:</td>
      <td><input type="text" oraField="address"/></td>
    </tr>
    <tr>
      <td>City:</td>
      <td><input type="text" oraField="city"/></td>
    </tr>
    <tr>
      <td>State:</td>
      <td><input type="text" oraField="state"/></td>
    </tr>
    <tr>
      <td>Zip:</td>
      <td><input type="text" oraField="zip"/></td>
    </tr>
    <tr>
      <td/>
      <td style="padding-top:15px;">
        <input type="button" value="Save" onClick="oraSubmitMap('SAVE');"/>
        <input type="button" value="Cancel" onClick="oraSubmitMap('CANCEL',false);"/>
      </td>
    </tr>
  </table>
</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

Save and Cancel buttons rendered:

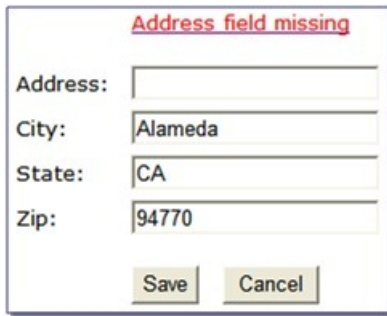


Figure 48: Save and Cancel buttons supporting oraSubmitMap

Include a Map Fragment

<oraInclude map='UI Map' prefixPath='xpath'/>

The oraInclude function is used to embed a map fragment within another UI map. Note that it is possible to use the include node within a map or a map fragment (but it is an error if you include a fragment within itself!).

- map: The name of a UI map fragment is required.
- prefixPath: Optionally specify an xpath prefix to be appended onto every included oraField, oraLabel, oraList, oraSelect valuePath and descPath, oraDownloadData, and oraUploadData attribute value defined within the included UI map fragment's HTML.

NOTE: The prefixPath functionality only applies to xpath attribute values when those values do not appear beneath an oraList attribute. Any xpath value within a table containing an oraList attribute will not be affected by a prefixPath.

- An example of a map fragment with two buttons, named 'F1-SaveCancelButtons'.

```
<input onClick = "oraSubmitMap('SAVE');" oraMdLabel="SAVE_BTN_LBL" class="oraButton" type="button"/>
>
<input onClick = "oraSubmitMap('CANCEL',false);" oraMdLabel="CANCEL_LBL" class="oraButton" type="button"/>
>
```

- An example of a map that includes the map fragment named 'F1-SaveCancelButtons'.

```
...
<tr>
  <td colspan="2" align="center">
<oraInclude map="F1-SaveCancelButtons"/>
  </td>
</tr>
...
```

Show Schema Default on Add

<action>ADD</action>

Default values within in the UI map's schema will be displayed on a UI map's input fields if an embedded <action> node has a value of 'ADD' or blank.

The schema default for the <description> element will be displayed:

```
<schema>
  <action/>
  <boGroup type="group">
    <key/>
    <description default="enter description here"/>
  </boGroup>
</schema>
<html>
```

```

<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Description  </td>
    <td><input oraField="boGroup/description"></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <action>ADD</action>
  <boGroup>
    <key/>
    <description/>
  </boGroup>
</root>
</xml>
</html>

```

Schema default value rendered:

Figure 49: Schema default value rendered for HTML input element

Configure a Chart

The following HTML attributes are used to configure a graphical representation of an XML list. The designer can control the type, size, position, and contents of the chart using the following attributes.

oraChart="type:pie, stacked, cluster, line, area, combo ;"

The **oraChart** attribute defines the type of graph to display and its general configuration. The set of configuration parameters available for this attribute are as follows:

Parameter	Values	Description
type:	pie	Defines the type of chart to display.
	stacked	Required
	cluster	
	line	
	area	
	combo	
showLegend	true	Defines if the chart should have a legend displayed.
	false	Optional (default is true)
legendPosition	left	Defines where the legend should appear.
	right	Optional (default is right)
	bottom	Setting position to left or right will automatically render it vertically.
	top	Setting position to top or bottom will automatically render it horizontally.
legendBorder	true	Defines if the legend should display with a border around it.
	false	Optional (default is false)

Parameter	Values	Description
depth	true	A value of true indicates a 3 dimensional depth for the chart.
	false	Optional (default false , which is a 2 dimensional chart)
animate	true	A value of true indicates that the graph should animate when displayed.
	false	Optional (default is true). When using large data sets, consider disabling animation.
dataCursor	on	A value of on enables hovering anywhere in the graph.
	off	Optional (default is off). It is not applicable to pie charts.
orientation	horizontal	Defines the chart orientation. This only applies to bar, line, area, combo charts. E.g., oraChart="type:cluster; orientation:horizontal" , defines horizontal cluster chart. Optional (default is vertical).

The **oraChartSeries** attribute defines the source information for the graph. There are 5 of these attributes:

- **oraChartSeries1**
- **oraChartSeries2**
- **oraChartSeries3**
- **oraChartSeries4**
- **oraChartSeries5**

Stacked charts support an unlimited number of series by continuing to add attributes **oraChartSeries6** and above, but beware of performance implications and memory limits when using an excessively high number of series.

All attributes are identical in format and accept the same parameters, as described below.

NOTE: All the charts require **oraChartSeries1**. Stacked, Cluster and Line charts may optionally include the additional series attributes (for multiple bars/lines).

If you define multiple series, then data must be provided for all series defined. The data amounts can be 0 (zero) but they have to be present in order for the chart to display correctly.

The set of configuration parameters available for the **oraChartSeriesN** attribute are:

Parameter	Values	Description
list:	XPath value	Defines the XPath to the list in the XML that contains the data to chart. Required
amount:	XPath value	Defines the XPath to the element in the XML list that contains the amount to chart. Required

Parameter	Values	Description
xaxis:	XPath value	<p>Defines the XPath to the element in the XML list that contains the x-axis data.</p> <p>Required for Stacked, Cluster, Line, Area and Combo charts.</p>
xaxisFormat:	date dateTime time localDate string	<p>Defines the x-axis data format.</p> <p>If it is date, dateTime or time then the value is presented in the format as defined on the user's display profile.</p> <p>In case of localDate or string the data is displayed as is with no special formatting.</p> <p>Optional (Default is date).</p>
label:	Text value	<p>Defines the label for the amount being charted.</p> <p>Either this setting or labelPath: must be defined.</p>
labelPath:	XPath value	<p>Defines the XPath to the element that provides the label for the amount being charted.</p> <p>Either this setting or label: must be defined.</p>
currency:	A valid Currency code	<p>Defines the currency code for the amount being charted.</p> <p>Optional.</p>
currencyPath:	XPath value	<p>Defines the XPath to the element that provides the currency code for the amount being charted.</p> <p>Optional.</p>
hoverText:	Text value	<p>Defines the hover text for the chart element.</p> <p>Optional (A default hover text is always available.) Ignored if hoverTextPath: is defined.</p> <p>The following substitution variables are available.</p> <ul style="list-style-type: none"> • \$label This will be replaced with the label text as determined by the label: or labelPath: setting. • \$amount This will be replaced with the amount text as specified by the amount: setting. • \$axis This will be replaced with the x-axis text. • \$% This will be replaced by the percentage "slice" of the pie or bar. • \$newline This will be force a line break.

Parameter	Values	Description
		<p>If the hover text defined contains any of the above values, they will be replaced by the equivalent text prior to being displayed.</p> <p>Example:</p> <pre>"hoverText:\$label\$newline\$amount"</pre>
hoverTextPath:	XPath value	<p>Defines the XPath to the element that provides the hover text for the chart element.</p> <p>The hover text in the XML can utilize all the substitution functionality described above in the hoverText: description.</p> <p>Optional.</p>
type:	bar line area	<p>This attribute is used for the combo chart type only. It defines how each series on the combo chart should be presented.</p> <p>The following example defines a combo chart where one series is rendered as bars and another one as area.</p> <pre>oraChart="type:combo;" oraChartSeries1="list</pre>
navOpt:	A valid Navigation Option code.	<p>Defines a navigation option to be activated when the chart element is clicked.</p> <p>Optional.</p>
navOptPath:	XPath value	<p>Defines the XPath to a navigation option to be activated when the chart element is clicked.</p> <p>Optional</p> <p>Note that both navOpt: and navOptPath: may be configured. The XPath navigation option is processed first. If a value is found it is used, otherwise the value in the navOpt: setting is used. This means that the HTML can define a "default" navigation option and a navigation option present in the XML document will override it.</p>
key:	XPath value	<p>Defines the XPath to an XML element in the series list that contains the key field data to be used in a navigation option.</p> <p>This is required if either navOpt: or navOptPath: is defined.</p> <p>NOTE: Only one key field can be configured for a navigation option.</p>
script:	A BPA script name	<p>Defines a BPA script to be activated when the chart element is clicked.</p> <p>Optional</p> <p>When a script is executed, all the elements from that list item will be made available to the script as temporary variables.</p>

Parameter	Values	Description
scriptPath:	XPath value	<p>Defines the XPath to a BPA script to be activated when the chart element is clicked.</p> <p>Optional</p> <p>Note that both script: and scriptPath: can be configured. The XPath script option is processed first. If a value is found it is used, otherwise the value in the script: setting is used. This means that the HTML can define a default script and a script present in the XML document will override it.</p>
color:	HTML Color code / RGB value	<p>Defines the color for the series. The format is valid HTML color code, e.g. green, red or black. All valid color names are defined in this link: http://www.w3schools.com/html/html_colornames.asp.</p> <p>Alternatively an RGB format may be used. (FF0000 is red, 00FF00 is green and 0000FF is blue)</p> <p>Optional (default colors applied)</p>
colorPath:	XPath value	<p>Defines the XPath to a color for the series. The valid format as described in the color: setting apply.</p> <p>Optional (default colors applied)</p>
pieColors:	HTML color code / RGB values	<p>Defines the colors for the pie series. Any number of HTML color codes or RGB color values can be specified, separated by spaces.</p> <p>Examples:</p> <pre>pieColors: red green black</pre> <pre>pieColors: FF0000 00FF00</pre> <p>Optional (default colors applied if the series exceeds the values specified)</p>

oraChartBroadcast="FIELD_NAME:XPath;"

A chart configured in a map zone can be set to broadcast portal context. An unlimited number of fields can be broadcast, configured as name/value pairs, as follows:

1. **FIELD_NAME:** the name of the portal context field to be broadcast.
2. **XPath:** the XML schema element from the same list item that corresponds to the user selected chart segment and contains the data value to be broadcast.

Graph Examples

- Sample of a pie chart configuration:

```
<html>
<head>
<title>Pie Chart</title>
</head>
<body>
```



```

<div style="width:100%; height:290px;"
  oraChart="type:pie;"
  oraChartSeries1="list:set; labelPath:date; amount:amount; "
  oraChartBroadcast="BILL_ID:billId;">
</div>

</body>

<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>
<amount>97.29</amount>
<billId>592211649442</billId>
  </set>
  <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<billId>592211649443</billId>
  </set>
</root>
</xml>
</html>

```

- A pie chart rendered for a single series:

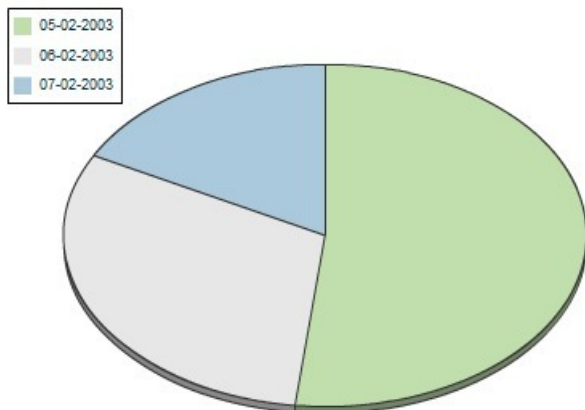


Figure 50: oraChart="type:pie;" rendered

- Sample of a line, cluster, or stacked graph configuration - each with two series:

```

<html>
<head>
<title>Stacked Chart</title>
</head>
<body>

<div style="width:100%; height=300px;"
  oraChart="type:line;"
  oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; "
  oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; "
  oraChartBroadcast="BILL_ID:billId;">
</div>

<div style="width:100%; height=300px;"
  oraChart="type:cluster;"
  oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; "
  oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; "
  oraChartBroadcast="BILL_ID:billId;">

```

```

</div>

<div style="width:100%; height=300px;"
oraChart="type:stacked;"
oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; "
oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; "
oraChartBroadcast="BILL_ID:billId;">
</div>

</body>

<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<balance>163.24</balance>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>
<amount>97.29</amount>
<balance>260.53</balance>
<billId>592211649442</billId>
  </set>
  <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<balance>314.91</balance>
<billId>592211649443</billId>
  </set>
</root>
</xml>
</html>

```

- Three types of chart rendered for two series each: line, cluster, and stacked.

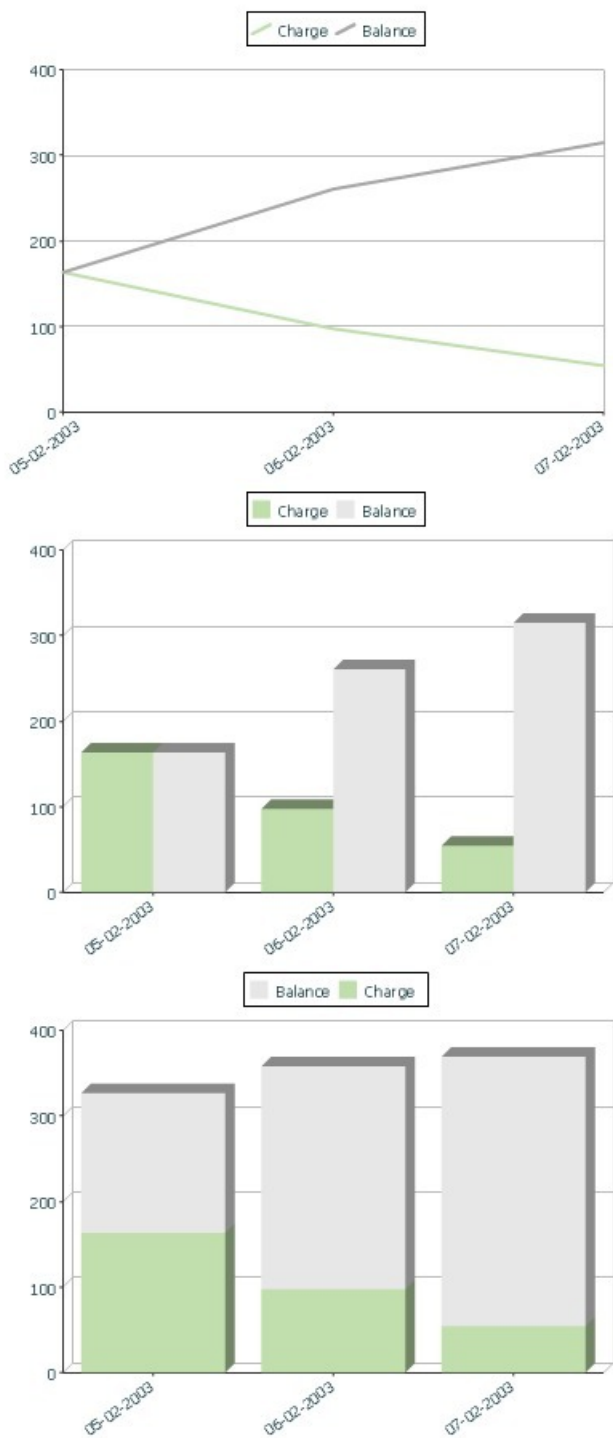


Figure 51: oraChart="type:line; type:cluster; type:stacked; " rendered

Upload and Download a CSV File

The following HTML attributes can be used to manage both an upload and a download between a list defined within the map's schema and a CSV (comma separated value) file.

The syntax is `oraUploadData="type:embed;path:list xpath;useLabels:true;showCount:true"`

Upload configuration requires you to name a CSV file to be uploaded, and an XML list as target. By convention, each CSV row will create a separate list instance. Each comma-separated field in the file will be uploaded as a separate element in the list. To embed an upload dialog within a map, the **oraUploadData** attribute must be associated with a container element such as a div, td, or span.

The optional **useLabels:true** value indicates that while parsing the upload CSV file, the headers are expected to be labels

NOTE: If you do not specify the **useLabels:true** value and the XML target element name is "camelCase" then the corresponding spreadsheet header should be title case with a space between words, e.g.;"Camel Case". Letters and special characters are not considered a different word, for example Address1 must be uploaded into the target XML element address1.

Specifying the optional **showCount:true** value will display the number of records uploaded.

CAUTION: If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to [BO Replace Action](#) for more information.

Sample of **oraUploadData="embed"** within a div element.

```
<html>
<head>
  <title>File Upload</title>
</head>
<body>

  <div oraUploadData="type:embed;path:myList" > </div>

</body>

<xml>
<root>
  <myList>
<id>838383930</id>
  <name>Janice Smith</name>
  </myList>
  <myList>
<id>737773730</id>
  <name>Bill McCollum</name>
  </myList>
</root>
</xml>
</html>
```

This file upload dialog will be embedded into the body of the page where the oraUploadData is defined.

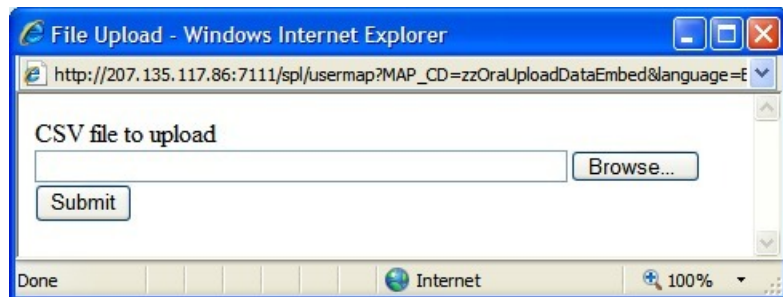


Figure 52: Embedded File Upload Dialogue

oraUploadData="type:popup;path:list xpath;useLabels:true;showOk:true;showCount:true"

Upload configuration requires you to name a CSV file to be uploaded, and an XML list as target. By convention, each CSV row will create a separate list instance. Each comma-separated field in the file will be uploaded as a separate element in the list. To upload a CSV file using a pop-up dialog, the oraUploadData attribute must be associated with an input element such as a button, text link, or image.

The optional useLabels:true value is used to indicate that while parsing the upload CSV file, the headers are expected to be labels

NOTE: If you do not specify the useLabels:true value and the XML target element name is "camelCase" then the corresponding spreadsheet header should be title case with a space between words, e.g., "Camel Case". Letters and special characters are not considered a different word, for example Address1 must be uploaded into the target XML element address1.

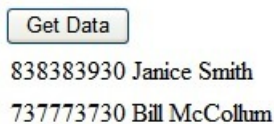
Specifying the optional showOk:true value will display an "Ok" button once the upload finishes. The popup will stay open until the button is pressed. Additionally, specifying the showCount:true value will display number of records uploaded.

CAUTION: If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to [BO Replace Action](#) for more information.

Sample of oraUploadData="popup" associated with a button:

```
<html>
<head>
  <title>File Upload</title>
</head>
<body>
  <input type="button" name="submitButton" oraUploadData="type:popup;path:myList;" value='Get Data'>
  <table oraList="myList">
    <tr/>
    <tr>
<td><span oraField="id"/></td>
<td><span oraField="name"/></td>
    </tr>
  </table>
</body>
<xml>
<root>
  <myList>
<id>838383930</id>
  <name>Janice Smith</name>
  </myList>
  <myList>
<id>737773730</id>
  <name>Bill McCollum</name>
  </myList>
</root>
</xml>
</html>
```

HTML Rendered:



Get Data

838383930 Janice Smith

737773730 Bill McCollum

Figure 53: oraUploadData attribute attached to the Get Data button

Pressing the "Get Data" button will launch a standard file upload dialogue (provided by Framework) as shown below.

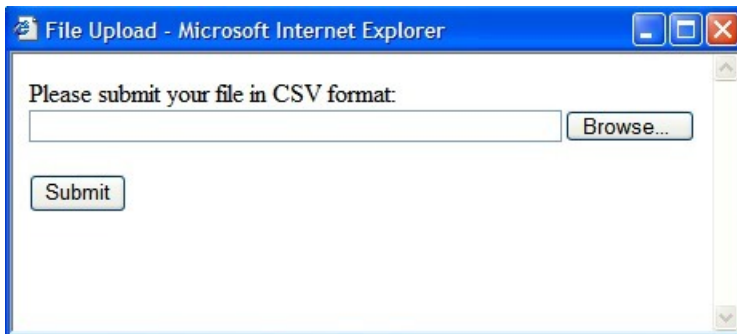


Figure 54: File Upload Dialogue Popup

oraDownloadData="list xpath"

Download configuration requires you to name an XML list to be downloaded. By convention, each list instance will represent a separate row in the created file. By default every element of the list will be comma separated in the file.

NOTE: The number formatting is based on the user profile setting. For localities where the decimal symbol is a comma, an implementation may configure a property setting (`spl.csv.delimiter.useFromDisplayProfile=true`) to cause the system to use a semicolon as the delimiter that separates the elements rather than a comma.

Sample of oraDownloadData.

```
<html>
<head>
<title>File Download</title></head>
<body>
<input type="button" name="downloadButton" oraDownloadData="myList" value="Download"/>
</body>
<xml>
<root>
  <myList>
    <id>881-990987</id>
    <name>John Mayweather</name>
  </myList>
  <myList>
    <id>229-765467</id>
    <name>Anna Mayweather</name>
  </myList>
  <myList>
    <id>943-890432</id>
    <name>Andrew Brewster</name>
  </myList>
</root>
</xml>
</html>
```

HTML Rendered:



Figure 55: oraDownloadData attribute attached to the Download button

Pressing the "Download" button will launch a standard file download dialogue (provided by Framework) as shown below.

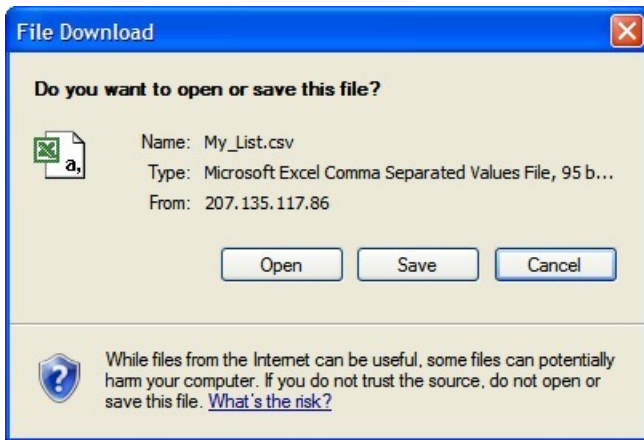


Figure 56: File Download Pop-up dialog

A successful download will result in a CSV file:

	A	B	C
1			
2	Id	Name	
3	881-99098	John Mayweather	
4	229-76546	Anna Mayweather	
5	943-89043	Andrew Brewster	
6			

Figure 57: CSV File created

To download data from a sub list use the attribute `oraDownloadDataInList` instead of `oraDownloadData`. The attribute `oraDownloadDataInList` will have the sub list name. The XPath of the sub list is used to pick data of the specific row from the parent list. Thus only the specific sub list is downloaded.

`oraDownloadDataUseLabels="true"`

The `oraDownloadDataUseLabels` attribute can be used in conjunction with the `oraDownloadData` attribute described above. Specify `oraDownloadDataUseLabels` if you want the generated CSV file to use the element labels for columns headers rather than element names.

Construct Portal Zone Map Fragments

Portal zones can reference a UI map for the zone header and filter area. This UI map is not a complete HTML document, but is instead configured as a UI Map "fragment". When constructing a zone map fragment you can reference the following substitution variables. Note that these variables will be dynamically populated at run time with information particular to the map's zone within the portal:

- `[$ZONEDESCRIPTION]` will be replaced with the zone description's text.
- `[$SEQUENCEID]` will be replaced with the zone's sequence id.
- `[$ZONENAME]` will be replaced with the name of the zone.
- `[$HELPTTEXT]` will be replaced with the help text entered on the zone transaction for that particular zone.
- `[$ZONEPARAMNAME]` will be replaced by the zone parameter's value or blank if it has not been specified.

CAUTION:

- Constructing a portal zone header is a highly technical exercise. Two delivered maps may be referenced as examples: F1-UIMapHeader and F1-ExplorerHeader.

- These maps make use of the [oraInclude](#) tag to incorporate HTML fragments for the header menu and framework actions. Refer to the zone type parameters for the UI Map fragments you should include in your HTML.
- If you wish to have the "help text" icon appear next to your zone description, you should have `id="title_[${SEQUENCEID}]"` on the `<td>` that contains your description.
- If it is necessary to encapsulate javascript within a UI Map fragment, it will be necessary to bound the javascript within a `![CDATA[]]` tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
/*![CDATA[ */
//
//javascript
//
/*]]> */
</script>
```

NOTE: If you wish to preserve the values of a filter input field, within a filter map fragment, for the framework 'Go Back' and 'Go Forward' functionality, you must associate the input field (text box, select, etc.) with a unique HTML id. Input field values associated with a unique id will be captured in the framework's 'memento'. The 'memento' is used to rebuild the input map when the portal zone is navigated to using the 'Go Back' or 'Go Forward' functionality.

NOTE: Many specialized functions exist to manipulate zone behavior, for example:

- `oraGetZoneSequence(zoneName)`: Use zone's code to retrieve its sequence number.
- `oraIsZoneCollapsed(sequenceId)`: Use zone's sequence to determine if collapsed.
- `oraHandleCollapse(seq)`: Collapse a zone.
- `oraHandleExpand(seq,refresh)`: Expand and/or refresh a zone.

All of these, and many more functions, are located within the javascript library [userMapSupport.js](#) described below.

NOTE: When executing `oraLoad` within a fragment UI map, and you need to execute a javascript function during page load (where the function invokes a business object, business service, or service script) you can use the special syntax "`oraLoad[${SEQUENCEID}]`". Refer to the [oraLoad](#) section for more information.

Example of `oraLoad[${SEQUENCEID}]` used within a function:

```
<script type="text/javascript">
function oraLoad[${SEQUENCEID}]() {
checkRebateClaimStatus();
}

function checkRebateClaimStatus() {
var work = id('analyticsFilterText[${SEQUENCEID}]',
document).cells[0].innerText.split(' ');
var rebateClaimId = work[work.length - 3];
id('rebateClaimId', document).value = rebateClaimId;
oraInvokeSS('C1-CheckRCSt','checkRebateClaimStatus', false);
var statusIndicator = id('statusInd', document).value;
if (statusIndicator == 'C1PE' || statusIndicator == 'C1ID') {
id('addRebateClaimLine', document).style.display = '';
} else {
id('addRebateClaimLine', document).style.display = 'none';
}
}
</script>
```

F1-ExplorerHeader rendered:



Figure 58: F1-ExplorerHeader rendered

Invoking a Business Object

The `oraInvokeBO` function may be used within a portal zone header or zone filter map. It is similar to the command described in [Invoke Schema Based Services](#) which allows for a business object to be invoked within the UI map's HTML. The syntax is as follows: `oraInvokeBO('BO name','xpath' | null,'action', null, [$SEQUENCEID], isHeader);`. Three extra arguments are required on top of what is already in the Invoke Schema Based Services section.

1. **null**. This must be specified as the fourth argument.
2. **[\$SEQUENCEID]**. This must be specified as the fifth argument.
3. *'isHeader'*. Specify **true** if the fragment is used within a portal zone header. Specify **false** if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeBO('CM-User','xmlGroup','read', null, [$SEQUENCEID], true)
```

Invoking a Business Service

The `oraInvokeBS` function may be used within a portal zone header or zone filter map. It is similar to the command described in [Invoke Schema Based Services](#) which allows for a business service to be invoked within the UI map's HTML. The syntax is as follows: `oraInvokeBS('BS name','xpath' | null, null, [$SEQUENCEID], isHeader);`. Three extra arguments are required on top of what is already in the Invoke Schema Based Services section.

1. **null**. This must be specified as the fourth argument.
2. **[\$SEQUENCEID]**. This must be specified as the fifth argument.
3. *'isHeader'*. Specify **true** if the fragment is used within a portal zone header. Specify **false** if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeBS('CM-UserSearch','xmlGroup', null, [$SEQUENCEID], true)
```

Invoking a Service Script

The `oraInvokeSS` function may be used within a portal zone header or zone filter map. It is similar to the command described in [Invoke Schema Based Services](#) which allows for a business service to be invoked within the UI map's HTML. The syntax is as follows: `oraInvokeSS('SS name','xpath' | null, null, [$SEQUENCEID], isHeader);`. Three extra arguments are required on top of what is already in the Invoke Schema Based Services section.

1. **null**. This must be specified as the fourth argument.
2. **[\$SEQUENCEID]**. This must be specified as the fifth argument.
3. *'isHeader'*. Specify **true** if the fragment is used within a portal zone header. Specify **false** if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeSS('UserSearch','xmlGroup', null, [$SEQUENCEID], true)
```

Hiding Portal Tabs

The product provides the ability to use javascript to hide a tab on the current portal based on some condition using the `oraAuthorizeTab` javascript API. This API accepts a function as a parameter and turns off the tab index indicated.

For example, the UI Map may have a function to turn off one or more tab indexes.:

```
function overrideTabIndex(index){
```

```
    if (index == 2) return false;
    if (index == 3) return false;
}
```

The javascript is referenced “on load”:

```
<body class="oraZoneMap"
onLoad="oraAuthorizeTabs(overrideTabIndex);">
```

Required JavaScript Libraries

All of the functionality described in this document depends on a pair of JavaScript libraries. If you are writing and executing your maps entirely within the UI map rendering framework - you do not need to manually insert the following libraries - the framework will insert them for you when the UI Map is rendered.

WARNING: When executing HTML outside of the framework you must include the following references explicitly within your HTML. In addition, the tool you use to render the HTML must have access to a physical copy of `privateUserMapSupport.js` for bind support.

src="privateUserMapSupport.js"

Your HTML document must reference this library to execute binding in a stand-alone environment.

WARNING: Referencing functions within this JavaScript library is dangerous - because these functions are owned by framework and they may be changed during version upgrade or via the normal patch process.

```
<script type="text/javascript" src="privateUserMapSupport.js"></script>
```

src="userMapSupport.js"

To take advantage of optional toolset features, you must reference this library.

NOTE: You can reference the functions within this JavaScript library to write custom functions within the UI map..

```
<script type="text/javascript" src="userMapSupport.js"></script>
```

onload="oraInitializeUserMap();"

To execute binding in a stand-alone environment, you must embed the following onload function into the `<body>` node.

```
<body onload="oraInitializeUserMap();">
```

UI Map Standards

Contents

- [Basic UI Map Templates](#)
- [Basic HTML and Styles](#)
- [Grids \(Tables of Data\)](#)
- [Action Buttons](#)
- [Available Styles](#)

Basic UI Map Templates

All UI Maps share the same basic structure regardless of placement (page area, zone, pop-up) or usage (display only, input).

Sample XML

All information in this document is based upon the following XML structure.

```
<xml>
  <root>
```

```

<address>123 Main St</address>
<city>Alameda</city>
<state>CA</state>
<zip>94770</zip>
<contactInformation>
  <type>Home Phone</type>
  <number>510-555-2287</number>
</contactInformation>
<contactInformation>
  <type>Cell Phone</type>
  <number>510-555-4285</number>
</contactInformation>
</root>
</xml>

```

Display Only UI Map

```

<html>
<head>
  <title oraMdLabel="ADDRESS_LBL"></title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body class="oraZoneMap">
<table cellpadding="4" cellspacing="4" width="100%">
  <colgroup>
    <col class="oraLabel oraTableLabel" />
    <col class="oraNormal oraTableData" />
  </colgroup>
  <tr>
    <td oraLabel="address"></td>
    <td oraField="address"></td>
  </tr>
  <tr>
    <td oraLabel="city"></td>
    <td oraField="city"></td>
  </tr>
  <tr>
    <td oraLabel="state"></td>
    <td oraField="state"></td>
  </tr>
  <tr>
    <td class="oraSectionEnd" oraLabel="zip"></td>
    <td class="oraSectionEnd" oraField="zip"></td>
  </tr>
  <tr>
    <td colspan="2" class="oraSectionHeader" oraMdLabel="CONTACT_LBL"></td>
  </tr>
  <tr>
    <td colspan="2" class="oraSectionStart oraEmbeddedTable">
      <table oraList="contactInformation" cellpadding="2">
        <thead>
          <tr>
            <th class="oraGridColumnHeader" nowrap="nowrap">
              <span oraLabel="contactInformation/type"></span>
            </th>
            <th class="oraGridColumnHeader" nowrap="nowrap">
              <span oraLabel="contactInformation/number"></span>
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td class="oraNormalAlt oraDisplayCell">
              <span oraField="type"></span>
            </td>
            <td class="oraNormal oraDisplayCell">
              <span oraField="number"></span>
            </td>
          </tr>
        </tbody>
      </table>
    </td>
  </tr>

```

```

        </td>
    </tr>
</table>
</body>
</xml>
<root>
  <address>123 Main St</address>
  <city>Alameda</city>
  <state>CA</state>
  <zip>94770</zip>
  <contactInformation>
    <type>Home Phone</type>
    <number>510-555-2287</number>
  </contactInformation>
  <contactInformation>
    <type>Cell Phone</type>
    <number>510-555-4285</number>
  </contactInformation>
</root>
</xml>
</html>

```

Input UI Map

```

<html>
<head>
  <title oraMdLabel="ADDRESS_LBL"></title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td><a href="" onclick="oraShowErrorAlert(); return false;"><span class="oraErrorText" oraErrorVar="
TEXT"></span></a></td>
  </tr>
</table>
<table cellspacing="4" width="100%">
  <colgroup>
    <col class="oraLabel oraTableLabel" />
    <col class="oraNormal oraTableData" />
  </colgroup>
  <tr>
    <td oraLabel="address"></td>
    <td><input type="text" oraField="address"/></td>
  </tr>
  <tr>
    <td oraLabel="city"></td>
    <td><input type="text" oraField="city"/></td>
  </tr>
  <tr>
    <td oraLabel="state"></td>
    <td><input type="text" oraField="state"/></td>
  </tr>
  <tr>
    <td oraLabel="zip"></td>
    <td><input type="text" oraField="zip"/></td>
  </tr>
  <tr>
    <td colspan="2" class="oraSectionHeader" oraMdLabel="CONTACT_LBL"></td>
  </tr>
  <tr>
    <td colspan="2" class="oraSectionStart oraEmbeddedTable">
      <table oraList="contactInformation" cellspacing="2">
        <thead >
          <tr>
            <th class="oraGridColumnHeaderButton"></th>
            <th class="oraGridColumnHeaderButton"></th>
            <th class="oraGridColumnHeader" nowrap="nowrap">
              <span oraLabel="contactInformation/type"></span>
            </th>
            <th class="oraGridColumnHeader" nowrap="nowrap">

```

```

        <span oraLabel="contactInformation/number"></span>
      </th>
    </tr>
  </thead >
  <tbody >
    <tr>
      <td oraType="addGridRow"></td>
      <td oraType="deleteGridRow"></td>
      <td>
        <input type="text" oraField="type"/>
      </td>
      <td>
        <input type="text" oraField="number"/>
      </td>
    </tr>
  </tbody>
</table>
</td>
</tr>
<tr>
  <td colspan="2" class="oraSectionStart oraEmbeddedTable">
    <table cellspacing="2">
      <tr>
        <td>
          <input class="oraButton" oraMdLabel="C1_SAVE_LBL" type="button" onClick="oraSubmitMap(
>
          </td>
          <td>
            <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button" onClick="oraSubmitMap(
>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
    <contactInformation>
      <type>Home Phone</type>
      <number>510-555-2287</number>
    </contactInformation>
    <contactInformation>
      <type>Cell Phone</type>
      <number>510-555-4285</number>
    </contactInformation>
  </root>
</xml>
</html>

```

Basic HTML and Styles

The basic templates introduced the standard HTML and styles used for UI Maps. These standards are described individually in the following sections.

Stylesheets

The styles to apply the standard look to the maps are all contained in stylesheets. These stylesheets should be included in all UI Maps.

```

...
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
...

```

Title

Each UI Map should have a <title> tag.

```
... <title oraMdLabel="ADDRESS_LBL"></title>
...
```

This will give the UI Map a descriptive title.

- If the UI Map is presented in a "pop-up", the title will be in the window title bar.
- If the UI Map is presented in the page area, the title will be added as a tag to the UI Map and will appear at the top of the UI Map.
- If the UI Map is presented as a zone map, it will be ignored. The <title> tag should still be included in the HTML as standard.

Zone Maps

When the map is presented in a zone as part of a portal, the UI Map should have a border so that the information is "contained" within the zone.

```
... <body class="oraZoneMap">
...
```

Page Area Maps vs Pop-Up Maps

The presentation of the UI Maps can vary from design to design. The following standards have been applied to decide when to use a Page Area UI Map and when to use a Pop-Up Map:

- If there are multiple UI Maps in the sequence, always use the Page Area.
- If the UI Map has many input fields, always use a Page Area.
- If the UI Map is a "confirmation" type dialog or only has one or two input fields, use a Pop-Up.

NOTE: The difference between "just a few input fields" and "many input fields" can be discretionary. The final decision should rest with the dialog designer.

Error Messages

Input maps have a ability to present error messages to the User.

```
...
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td><a href="" onclick="oraShowErrorAlert(); return false;"><span class="oraErrorText" oraErrorVar="ERRMSG-
TEXT"></span></a></td>
  </tr>
</table>
...
```

This HTML structure provides the provides the necessary elements and functions to display errors to the User. It should be directly after the <body> tag. When there is no error, nothing will be visible on the UI Map. It will be made visible if an error occurs and the UI Map is re-presented to the User. Clicking on the link (when visible) will result in a pop-up alert appearing with the long error message text.

Standard Layout and Styles

The information is presented on the UI Map by using a <table> to organize the information in rows and columns.

```
...
<table cellspacing="4" width="100%">
<colgroup>
```

```

<col class="oraLabel oraTableLabel" />
  <col class="oraNormal oraTableData" />
</colgroup>
...

```

The `<colgroup>` and `<col>` tags allow for the application of classes to the columns (the label is in the first column and the data is in the second column.). Using these tags mean that the class attribute (to apply styles) does not need to be defined on every `<td>`.

Grids (Tables of Data)

A UI Map could contain information that is best presented as a grid. These are referred to as "Embedded Tables". The embedded table can be used to display information or input information.

Example Embedded Table HTML

The embedded table will be included within a row (`<tr>`) of the basic layout:

```

...
<tr>
  <td colspan="2" class="oraEmbeddedTable">
    <table oraList="contactInformation" cellspacing="2">
      <thead >
        <tr>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/type"></span>
          </th>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/number"></span>
          </th>
        </tr>
      </thead >
      <tbody>
        <tr>
          <td class="oraNormalAlt oraDisplayCell">
            <span oraField="type"></span>
          </td>
          <td class="oraNormal oraDisplayCell">
            <span oraField="number"></span>
          </td>
        </tr>
      </tbody>
    </table>
  </td>
</tr>
...
<xml>
  <root>
    <address> 123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
    <contactInformation>
      <type>Home Phone</type>
      <number>510-555-2287</number>
    </contactInformation>
    <contactInformation>
      <type>Cell Phone</type>
      <number>510-555-4285</number>
    </contactInformation>
  </root>
</xml>

```

Embedding the Table

The embedded table is included within the overall table structure. The `colspan` attribute ensures that the embedded table can span the standard two columns of the overall layout table.

```

...

```

```

<tr>
  <td colspan="2" class="oraEmbeddedTable">
    ...
    ...
    ...
  </td>
</tr>
...

```

Embedded Table Structure

The embedded table is very similar to the basic layout table.

```

...
<table oraList="contactInformation" cellspacing="2">
<thead>
  ...
  ...
</thead>
<tbody>
  ...
  ...
</tbody>
</table>
...

```

- The <table> tag has a slightly smaller cellspacing and it defines the "list" element contained in the XML that will be used to provide the data.
- The <thead> element is used to give the embedded table headings for the columns.
- The <tbody> element is the element that will be repeated for each referenced "list" element in the XML. In the previous example, there are two "contactInformation" list elements, so the displayed embedded table will have two rows.

Column Headings

Embedded tables should have headings for the displayed columns. The <thead> tag defines these.

```

...
<thead>
  <tr>
    <th class="oraGridColumnHeader" nowrap="nowrap">
      <span oraLabel="contactInformation/type"></span>
    </th>
    <th class="oraGridColumnHeader" nowrap="nowrap">
      <span oraLabel="contactInformation/number"></span>
    </th>
  </tr>
</thead>
...

```

The "nowrap" attribute prevent the column heading from taking multiples lines. If multiples lines are required, the "nowrap" may be removed.

Input Fields

Embedded tables may be used for input as well as display only. The framework provides a convenient control to assist in the creation of editable embedded tables.

```

...
<tr>
  <td colspan="2" class="oraEmbeddedTable">
    <table oraList="contactInformation" cellspacing="2">
      <thead >
        <tr>
          <th class="oraGridColumnHeaderButton"></th>
          <th class="oraGridColumnHeaderButton"></th>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/type"></span>
          </th>
          <th class="oraGridColumnHeader" nowrap="nowrap">

```



```

        <span oraLabel="contactInformation/number"></span>
      </th>
    </tr>
  </thead >
  <tbody>
    <tr>
      <td oraType="addGridRow"></td>
      <td oraType="deleteGridRow"></td>
      <td>
        <input type="text" oraField="type"/>
      </td>
      <td>
        <input type="text" oraField="number"/>
      </td>
    </tr>
  </tbody>
</table>
</td>
</tr>
...

```

There are two new columns added to the input embedded table.

- oraType="addGridRow" will add a "+" button to the row. This will allow the User to add an additional row to the existing grid.
- oraType="deleteGridRow" will add a "-" button to the row. This will allow the User to delete an existing row from the grid.

NOTE: The <thead> tag also requires these two new columns to be added.

These controls are, as standard, placed at the beginning of the row in the order shown. Either of the controls may be omitted if required (if, for example, Users are not permitted to delete information).

The presence of either of these controls will activate the "empty list" process. This means that if the XML has no data for the "list" specified, the input grid will display with an empty row ready for the input of new information.

Action Buttons

Example Action Button HTML

Action buttons are used to perform some specified function from the UI Map. The actions are as varied as the information being displayed/updated. Below are two common examples:

- Save. Normally used on an Input UI Map to allow a User to save any changes they have made.
- Cancel. Normally used on an Input UI Map to allow a User to cancel changes in progress.

```

...
<tr>
  <td colspan="2" class=" oraEmbeddedTable">
    <table cellpadding="2">
      <tr>
        <td>
          <input class="oraButton" oraMdLabel="C1_SAVE_LBL" type="button" onClick="oraSubmitMap('OK
>
          </td>
        <td>
          <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button" onClick="oraSubmitMap('CAN
>
        </td>
      </tr>
    </table>
  </td>
</tr>
...

```

Button Standards

The following points highlight some standards related to buttons.

- Buttons are included as an embedded table.
- Buttons should be grouped together. They should not be placed in different areas of the UI Map.
- The location of the buttons depends mainly on the type of UI Map.
 - Display Only UI Maps should have a Record Actions section in the upper right section of the UI map.
 - Input UI Maps should have the buttons at the foot of the UI Map (after all input fields).

Available Styles

Styles are all contained in the referenced CSS stylesheets. They are applied by the HTML "class" attribute. The actual style settings used are NOT documented here as they may be adjusted. This section only specifies when a particular style should be used.

NOTE: The "class" attribute may reference more than one style (class="oraLabel oraSectionEnd")

Style	Comments	Example
oraButton	Applied to <input> elements where the type is button.	<pre>... <input class="oraButton" oraMdLabel="CANCEL" type="button" value="CANCEL" /> ...</pre>
oraDisplayCell	Applied to the <td> tag of an embedded table. It defines how the table cell looks (not the data contained inside the cell).	<pre>... <td class="oraDisplayCell"> </td> ...</pre>
oraEmbeddedTable	Applied to the <td> tag that will contain the embedded table.	<pre>... <tr> <td colspan="2" class=" oraEmbeddedTable"> <table cellspacing="2"> </table> </td> </tr> ...</pre>
oraError	This style is applied to elements that are identified as "error elements". Refer to Display Errors for more information. NOTE: This style is not normally applied directly in the UI Map HTML	
oraErrorText	This style is applied to the elements concerned with error messages.	<pre>... <table width="100%" cellpadding="12"> <tr class="oraErrorText"> <td> span></td> </tr> </table> ...</pre>
oraGridColumnHeader	This style is applied to the <td> tags that define column headers within embedded table.	<pre>... <thead > <tr> <th class="oraGridColumnHeaderButton">< <th class="oraGridColumnHeaderButton"><</pre>

Style	Comments	Example
oraGridColumnHeaderButton	<p>This style is applied to the <td> tags that define the column headers for the "+" and "-" buttons used on editable embedded tables.</p>	<pre> <th class="oraGridColumnHeader" nowrap= <span oraLabel="contactInformation"/ </th> <th class="oraGridColumnHeader" nowrap= <span oraLabel="contactInformation"/ </th> </tr> </thead > ... </pre>
oraInput	<p>This style is applied to input fields:</p> <ul style="list-style-type: none"> • <input type="text"> • <input type="checkbox"> • <select> • <textarea> <p>NOTE: This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".</p>	<pre> ... <input type="text" class="oraInput" oraField= ... </pre>
oraInputMoney	<p>This style is applied to input fields:</p> <ul style="list-style-type: none"> • <input type="text"> • <select> (rare) • <textarea> (not recommended) <p>NOTE: This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".</p>	<pre> ... <input type="text" class="oraInputMoney" oraField= ... </pre>
oraInputNumber	<p>This style is applied to input fields:</p> <ul style="list-style-type: none"> • <input type="text"> • <select> (rare) • <textarea> (not recommended) <p>NOTE: This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".</p>	<pre> ... <input type="text" class="oraInputNumber" oraField= ... </pre>
oraLabel	<p>This style is applied to standard label fields that are right aligned.</p>	<pre> ... <td class="oraLabel" oraLabel="address"></td> ... </pre>

Style	Comments	Example
	<p>NOTE: This can normally be omitted as it is applied by the <col> tag.</p>	
oraLabelAlt	This style is applied to standard label fields only if it is desired to have the label aligned to the left.	<pre>... <td class="oraLabelAlt" oraLabel="address"></td> ...</pre>
oraLabelCenter	This style is applied to standard label fields only if it is desired to have the label aligned in the center of the cell.	<pre>... <td class="oraLabelCenter" oraLabel="address"></td> ...</pre>
oraLink	This style is applied to foreign key references (links). This is automatically added by the UI Map framework but can also be used manually if desired.	<pre>... <td class="oraLabel"> Go </td> ...</pre>
oraNormal	This style is applied to standard data fields (display only) that are left aligned. <p>NOTE: This can normally be omitted as it is applied by the <col> tag.</p>	<pre>... <td class="oraNormal" oraField="address"></td> ...</pre>
oraNormalAlt	This style is applied to standard data fields (display only) only if it is desired to have the data aligned to the right.	<pre>... <td class=" oraNormalAlt" oraField="address"></td> ...</pre>
oraNormalCenter	This style is applied to standard data fields (display only) only if it is desired to have the data aligned in the center of the cell.	<pre>... <td class=" oraNormalCenter" oraField="address"> ...</pre>
oraPageTitle	This style is applied to the element that contains the page title. <p>NOTE: This style is not normally applied directly in the UI Map HTML. The is created by the UI Map framework when the UI Map is displayed in the page area.</p>	<pre>... <span class=" oraPageTitle" oraMdField="PAGE_TIT ...</pre>
oraSectionEnd	This style is applied to the <td> tags at the end of a "section" (group of elements). It provides some space to separate the section from the following information. <p>NOTE: The style must be applied to both <td> tags or the label may be misaligned with the data/input.</p>	<pre>... <tr> <td class="oraSectionEnd" oraLabel="zip"></td> <td class="oraSectionEnd" oraField="zip"></td> </tr> ...</pre>
oraSectionHeader	This style is applied to the <td> tag used to give a heading for a section within the information being displayed. It does not provide spacing before or after itself. The oraSectionStart and oraSectionEnd classes are used for this. <p>NOTE: The section header should span both the label column and the data column.</p>	<pre>... <tr> <td class="oraSectionHeader" colspan="2" ora </tr> ...</pre>
oraSectionStart	This style is applied to the <td> tags at the start of a "section" (group of elements). It provides some	<pre>... <tr></pre>

Style	Comments	Example
	<p>space to separate the section from the previous information (often a section header).</p> <p>NOTE: The style must be applied to both <td> tags or the label may be misaligned with the data/input.</p>	<pre><td class="oraSectionStart" oraLabel="zip"> <td class="oraSectionStart" oraField="zip"> </tr> ...</pre>
oraTableData	<p>This style is applied to the <col> tag for the data column of the main table (second column). It is used to provide a percentage width for the horizontal space to be used for the information.</p>	<pre>... <colgroup> <col class="oraLabel oraTableLabel" /> <col class="oraNormal oraTableData" /> </colgroup> ...</pre>
oraTableLabel	<p>This style is applied to the <col> tag for the label column of the main table (first column). It is used to provide a percentage width for the horizontal space to be used for the labels.</p>	<pre>... <colgroup> <col class="oraLabel oraTableLabel" /> <col class="oraNormal oraTableData" /> </colgroup> ...</pre>
oraTinyText	<p>This style is typically applied directly beneath an <input> tag to provide information or a hint to the user concerning information relevant to the input. For example, name or address format.</p>	<pre>... <tr> <td oraLabel="address"></td> <td> <input type="text" oraField="address" <div class="oraTinyText" oraFie </td> </tr> ...</pre>
oraZoneMap	<p>This style is used applied when the UI Map is to be displayed as a zone on a portal.</p>	<pre>... <body class="oraZoneMap"> ...</pre>

Ensuring Unique Element IDs for UI Maps

The following describes how to modify JavaScript code to ensure the proper rendering of unique element IDs for UI Maps.

The modification is required only for code that renders HTML using a `getElementById()` (or similar) function to generate list IDs and avoid account verification or related errors.

The following sample snippet contains the necessary modifications:

```
...
function getElementsFromList(namePrefix) {
  var ret = [];
  var elements = document.getElementsByTagName("INPUT");
  for(var i=0;i<elements.length;i++) {
    var elemID = elements[i].id;
    if((id) && (id.startsWith(namePrefix + '_'))) {
      ret.push(elements[i]);
    }
  }
  ...
return ret;
}
```

Since IDs aren't necessarily unique in generated UI Map IDs, the code shown above ensures uniqueness at runtime by appending an underscore and row number (e.g., `myField_1`, `myField_2`) for proper handling by Framework in the rendered HTML, while still allowing you to reference the unmodified IDs contained in the generated UI Map.

A switch in the `spl.properties` file also permits you to disable the generation of unique IDs for elements in a grid (as described below), though, for standards compliance reasons, it is highly recommended that this switch be left at its default value.

```
Property Name: spl.runtime.compatibility.uiMapDisableGenerateUniqueHtmlIDs
File Name: spl.properties (under web project in FW)
Default Value: false
Accepted Values: true or false
Description: This property controls the generation of unique IDs for all input elements inside a list. When t
compliant.
```

Maintaining Managed Content

The Managed Content maintenance object is used to store content such as XSL files used to create vector charts, JavaScript include files, and CSS files. These files may then be maintained in the same manner as the HTML in UI Maps.

The topics in this section describe the Managed Content portal.

Managed Content - Main

This page is used to define basic information about the content. Open this page using **Admin > System > Managed Content**.

Description of Page

Enter a unique name for the content in the **Managed Content** field.

Owner indicates if the content is owned by the base package or by your implementation.

Use **Managed Content Type** to indicate the type of content, for example, XSLT or JavaScript.

Enter a **Description**.

Use the **Detailed Description** to describe in detail how this map is used.

Managed Content - Schema

This page is used to create and maintain the managed content. Open this page using **Admin > System > Managed Content** and then navigate to the **Schema** tab.

Description of Page

The General Information zone displays the main attributes of the content. The Editor zone allows you to edit the content.

Data Areas

The data area has no business purpose other than to provide a common schema location for re-used schema structures. It exists solely to help eliminate redundant element declaration. For example, if you have multiple schemas that share a common structure, you can set up a stand-alone data area schema for the common elements and then include it in each of the other schemas.

Be aware that a stand-alone data area can hold elements that are mapped to true fields. For example, you might have 50 different types of field activities and all might share a common set of elements to identify where and when the activity should take place. It would be wise to declare the elements that are common for all in a stand-alone data area and then include it in the 50 field activity business objects.

It's strongly recommended that you take advantage of stand-alone data areas to avoid redundant data definition!

CAUTION: Dynamic inclusion! When the system renders a schema, all schemas included within it are expanded real-time. This means that any change you make to a data area will take effect immediately within all schemas it is referenced within.

NOTE:

Schema Tips. The data area page includes a special Schema Tips zone that provides a link to launch help topics related to the [Advanced Schema Topics](#) help in one click.

Data areas may be included in a business object that does not define a full UI map for display or input. Rather, it is using auto-rendering by defining UI attributes in its schema and via UI hints.

NOTE: View UI Rendering. A context sensitive "View UI Rendering" zone appears on this page. It may be used for a data area that is part of a business object that is using auto-rendering for the display and input maps. The buttons allow you to view the rendered UI for the segment of the schema that is defined by the data area.

Defining Data Areas

The topics in this section describe how to maintain Data Areas.

Data Area - Main

Use this page to define basic information about a data area. Open this page using **Admin > System > Data Area**.

Description of Page

Enter a unique **Data Area** name and **Description**. Use the **Detailed Description** to describe what this data area defines in detail. **Owner** indicates if the data area is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new data area, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Click the **View Schema** to view the data area's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

To extend another data area, reference that data area in the **Extended Data Area** field. By extending a data area you can add additional elements to a base product data area.

Here's an example of an extended data area:

- The product releases with data area A, which contains elements a, b, and c.
- Your implementation creates data area CM-A, which contains element z, and references data area A as the extended data area.
- At run time, everywhere data area A is included it will contain elements a, b, c, and z.

Where Used

Follow this link to open the data dictionary to view the tables that reference [FI_DATA_AREA](#).

Data Area - Schema

Use this page to maintain a Data Area's schema and to see where the data area is used in the system. Open this page using **Admin > System > Data Area** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the main attributes of the data area.

The [Schema Designer](#) zone allows you to edit the data area's schema. The purpose of the schema is to describe the structure and elements of the data area.

FASTPATH: Refer to [Schema Syntax](#) and [UI Hint syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the dashboard provides links to launch these help topics directly.

NOTE: View UI Rendering. A context sensitive "View UI Rendering" zone is associated with this page. The zone is useful for data areas that are to be included in [business objects that define the user interface detail](#) using schema attributes and UI Hints. The buttons allow you to view the automatically rendered display and input maps.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Advanced Schema Topics

The topics in this section describe some advanced information related to schemas used for business objects, business services, service scripts and UI maps.

Schema Nodes and Attributes

For business object definition, the purpose of the schema is to create a link between the schema and a maintenance object. For business service definition you are specifying the link between the schema and a service (either a general service, search service, or a maintenance object service). For service script definition, you are defining the API for passing information to and from the script. The following documentation is a complete list of the XML nodes and attributes available to you when you construct a schema.

Contents

- [The Four Element Types](#)
- [The Data Type of a Field Element](#)
- [Referencing Other Elements](#)
- [Standard Time Considerations](#)
- [The Mapping Attributes](#)
- [Descriptive Attributes](#)
- [Schema Constants](#)
- [Defaulting and System Variables](#)
- [The Flattening Nodes and Attributes](#)
- [Search Zone](#)
- [Extend Security for Service Script](#)
- [Overriding Action for a Business Service](#)
- [Specifying searchBy for a Search Service](#)
- [Including Other Schemas](#)
- [Compatibility Attributes](#)

The Four Element Types

A schema element can be one of four different structure types. Note that there are two classes of element types: the structural nodes group and list, and the data containing nodes of field and raw.

Mnemonic	Valid Values	Description	Examples
type=	“field”	The field type is the default type for any element not explicitly labeled as something other than a field. Therefore, you virtually never have to explicitly label an element as a field. Note that a field element, unlike group or list, will contain information in its nodes - rather than other nodes.	
	“group”	<p>The group element is typically a structural element of the schema only, in which case it has no mapping.</p> <p>Note that when grouping several elements that are all used to map an XML structure of a CLOB / XML field of a business object driven record, the mapping may be at the group level.</p>	<p>Example where a group is used to create a structure</p> <pre><schema> <input type="group" <userId/> </input> <output type="group"> <firstName/> <lastName/> </output> </schema></pre> <p>Example where the group includes the mapping:</p> <pre><parameters type="group" mapXML="BO_ > <numberOfDays mdField="F1_NBR_DAYS > <frequency mdField="F1_FREQUENCY" ></pre>
	“list”	The list node is structural node like the group node. The only difference is that the list structure has the ability to repeat multiple times in an XML document.	<p>Example of a schema with a list:</p> <pre><schema> <statesList type="list"> <state isPrimeKey="true" / > <description/> </statesList> </schema></pre> <p>Example of a schema with a list:</p> <pre><xml> <statesList> <state>AK</state> <description>Alaska</ description> </statesList> <statesList> <state>AL</state> <description>Alabama</ description> </statesList> . . . </xml></pre>
	“raw”	The raw data type is used to capture a chunk of raw text that doesn't have any inherent structure associated with it that .	<pre><sendDetail type="raw" / ></pre> <p>Example of an XML instance for the above schema:</p> <pre><sendDetail> <messageInfo> <senderAddress>123 W. Main St, C senderAddress></pre>

Mnemonic	Valid Values	Description	Examples
			<pre><corpZone>3A</corpZone> </messageInfo> </sendDetail></pre>

The Data Type of a Field Element

Of the four different element types, only a field can have a data type.

Mnemonic	Valid Values	Description	Examples
dataType=	"string"	By default, a field element is a string. Therefore, there is no requirement to specify the string data type.	<pre><schema> <custName dataType="string" / > </schema></pre>
	"number"	<p>Defines an element that is a number.</p> <p>NOTE: UI hints include a setting to Suppress Automatic Number Formatting.</p> <p>NOTE: Use currencyRef attribute for auto-display of currency symbol that is associated with the referenced currency code. The currency decimal positions are ignored by this formatting allowing you to display a currency symbol for a unit rate with many decimals.</p>	<p>Examples</p> <pre><schema> <count dataType="number" / > </schema></pre> <pre><schema> <taxRate dataType="number" curren > </schema></pre>
	"money" Optional additional attributes currencyRef="element name"	<p>Defines an element that represents a monetary amount.</p> <p>The currency reference is optional and if left blank the installation currency will be used. Automatic formatting and validation to be applied based on the currency. For example, the currency symbol will be shown when auto-rendering. In addition, the number of decimal places must not exceed the valid number defined for the currency.</p> <p>NOTE: Refer to Referencing Other Elements for supported</p>	<pre><schema> <currency default="USD" suppress= > <balance dataType="money" curren > </schema></pre>

Mnemonic	Valid Values	Description	Examples
		syntax for referring to other elements.	
“lookup”	Required additional attribute lookup=“field name”	Defines an element that has valid values defined using a lookup. The lookup field is required.	<pre><schema> <status dataType="lookup" lookup=" > </schema></pre>
“lookupBO”	Required additional attribute lookupBO=“bo name”	Defines an element that has valid values defined using an extendable lookup. The extendable lookup’s business object is required.	<pre><schema> <category dataType="lookupBO" look BusinessCategory"/> </schema></pre>
“boolean”		Defines an element that has values of “Y” and “N”.	<pre><schema> <allowsEdit dataType="boolean"/ > </schema></pre>
“date”		Defines an element that represents a date.	<pre><schema> <startDate dataType="date"/ > </schema></pre>
“dateTime”		Defines an element that represents a date and time. NOTE: Refer to Standard Time Considerations for additional configuration available for date / time fields that represent standard time.	<pre><schema> <startDateTime dataType="dateTime" > </schema></pre>
	dataType="time”	Defines an element that represents a time.	<pre><schema> <startTime dataType="time"/ > </schema></pre>

Referencing Other Elements

There are several attributes that allow for a reference to another element in the same schema. The supported syntax of the XPath reference is the same in every case. This section provides examples using the default reference attribute (**defaultRef**).

Reference a sibling element:

```
<schema>
  <id mapField="ACCT_ID" required="true"/>
  <altId defaultRef="id" required="true"/>
</schema>
```

Reference an element in a higher group:

```
<schema>
  <id mapField="ACCT_ID" required="true"/>
  <msgInfo type="group" mapXML="XML_FIELD">
    <altId defaultRef="../id" required="true"/>
  </msgInfo>
</schema>
```

Reference an element in a lower group:

```
<schema>
  <id mapField="ACCT_ID" defaultRef="msgInfo/altId" required="true"/>
  <msgInfo type="group" mapXML="XML_FIELD">
```

```
<altId required="true"/>
</msgInfo>
</schema>
```

Reference an element in another group:

```
<schema>
<acctInfo type="group">
  <id mapField="ACCT_ID" required="true"/>
</acctInfo>
<msgInfo type="group" mapXML="XML_FIELD">
  <altId defaultRef="../../acctInfo/altId" required="true"/>
</msgInfo>
</schema>
```

Standard Time Considerations

Most date / time fields represent “legal” time such that if a time zone changes their clocks for winter and summer time, the date / time field captures the current observed time. However, some date / time fields should always be captured in standard time to avoid confusion / ambiguity. A good example is a date and time related to detailed interval data.

When defining an element with **dataType**=“**dateTime**”, you may optionally configure **stdTime**=“**true**” indicating that data captured in the element always represents standard time in the 'base' time zone. The 'base' time zone is specified on the [Installation options](#).

NOTE: If an element is mapped to a table / field with a Standard Time Type of **Physical**, then **stdTime**=“**true**” is implied. Refer to [Table / Field](#) for more information.

Example:

```
<schema>
<startTime dataType="Time" stdTime="true"/>
</schema>
```

If the time zone that represents the date / time field is not the installation time zone, use the optional setting **stdTimeRef**=“XPath to time zone element” on a date / time element to indicate that the element represents standard time and indicates the time zone to use. Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements. .

Example:

```
<schema>
<alternateTimeZone fkRef="F1-TZONE" suppress="true"/>
<startDateTime dataType="dateTime" stdTimeRef="alternateTimeZone"/>
</schema>
```

NOTE: If an element is mapped to a table / field with a Standard Time Type of **Referenced**, then **stdTime**=“XPath” is implied. Refer to [Table / Field](#) for more information.

NOTE: When schema elements are captured in standard time the UI map supports HTML notation to automatically display the data applying a daylight savings time / summer time correction. Refer to the HTML attribute [oraType="dateTime; stdTime:true"](#) for more information.

There may be cases where the date / time is captured as standard time in one time zone, but should be displayed using a different time zone. In this case, the attribute **displayRef**=“XPath” may be used in addition to the appropriate attribute that identifies the time zone that the data is capture in. Refer to [Referencing Other Elements](#) for supported syntax.

```
<schema>
<displayTimeZone fkRef="F1-TZONE" suppress="true"/>
<startDateTime dataType="dateTime" stdTime="true" displayRef="displayTimeZone"/>
</schema>
```

The Mapping Attributes

When constructing your schema, you can choose from one of the following mapping attributes.

Mnemonic	Valid Values	Description	Examples
mapField=	"field name"	In the case of a business object, the mapField attribute is used to identify the database column the element is related to. For business service schemas, the mapField= attribute is used to link a schema element with a service element.	<pre><schema> <factId mapField="FACT_ID" / > </schema></pre>
mapChild=	"table name"	<p>The mapChild= attribute is used only for business object mapping. It is used in two ways:</p> <ul style="list-style-type: none"> First, to create a list in the business object that corresponds to a child table of an MO. Second, you can use mapChild to identify the child table a flattened field lives in. For more information on flattening, refer to flattening section below. 	<p>Example of a list within a child table in a BO:</p> <pre><persons type="list" mapChild="CI_AC <personId mapField="PER_ID" / > </persons></pre>
mapList=	"list name"	<p>The mapList attribute is used only for business service mapping. It is used in two ways:</p> <ul style="list-style-type: none"> First, to create a list in the business service that corresponds to a list in the service. Second, you can use mapList to identify the list a flattened field lives in. For more information on flattening, refer to flattening section below. Both of the above uses of mapList within a complete business service schema. 	<p>Example of a list within a business service:</p> <pre><selectList type="list" mapList="DE <value mapField="COL_VALUE"> <row mapList="DE_VAL"> <SEQNO is="2"/> </row> </value> </selectList></pre>
mapXML=	"field name"	The mapXML attribute is typically used to map XML structures into a large character / XML field of the service. Note that when you use mapXML to map either a list or group structure (type="list" or type="group") you don't have to map all the child elements within	<pre><enrollmentRequest type="group" map <messageID/> <sender/> </enrollmentRequest> <enrollmentKey mapXML="CASE_CLOB" / > <enrollmentInfo type="list" mapChild <sequence mapField="CHILD_SEQ" / ></pre>

Mnemonic	Valid Values	Description	Examples
		the structure. It is also possible to map list elements to a large character field associated with the list child.	<pre><name mapXML="CASE_CHILD_CLOB" /> > <value mapXML="CASE_CHILD_CLOB" /> > </enrollmentInfo></pre>

isPrimeKey="true"

You are required to specify a primary key for a list defined within a mapped XML element (**type="list" mapXML="CLOB"**). The primary key is used by the framework to determine whether a list element add, update, or delete is required during a business object update.

NOTE: You do not need to specify the prime key for a business object list mapped to maintenance object list. When a physical list is mapped, the prime key is derived from existing physical meta-data.

```
<questionnaire type="list" mapXML="CASE_CLOB">
  <question isPrimeKey="true"/>
  <answer/>
</questionnaire>
```

orderBy="xpath asc|desc, xpath asc|desc"

By default, a list defined within a mapped XML element (**type="list" mapXML="CLOB"**) is sorted by the first element of the list. A different sort order may be specified using the **orderBy** attribute. The attribute value is a comma separated list of field XPathS (relative to the list element) with an optional sort order [**asc,desc**] (ascending is default).

NOTE: This attribute is only available for lists mapped as XML within business objects.

```
<questionnaire type="list" orderBy="page/section, page/sequence" mapXML="CASE_CLOB">
  <question isPrimeKey="true"/>
  <answer/>
  <page type="group">
    <section/>
    <sequence/>
  </page>
</questionnaire>
```

Descriptive Attributes

The following attributes can be used to describe a schema element and provide additional configuration related to the element. Typically, these attributes are useful for field elements only.

<!-- comment -->

You can add a comment to your schema (always a good idea!) by using the special open and close characters: <!-- and -->.

```
<schema>
  <!-- This schema is used to capture business information only, please refer to the 'HUMAN' BO for person in
  >
</schema>
```

description="text"

The description of the element is not to be confused with the label. The description is meant to be an internal description of the element to help a reader of the schema to understand the business reason for the element.

```
<schema>
  <active type="boolean" description="active account" label="Active"/>
</schema>
```

label="text"

The label of an element is meant to be a short bit of verbiage that would typically precede the element in a user interface.

```
<schema>
  <active type="boolean" description="active account" label="Active"/>
</schema>
```

required="true"

You can require the existing of a node during object interaction.

NOTE: Note on [included](#) schemas: The required="true" attribute will not be processed on business object and business service schemas when they are included within a service script schema. This is to support the ability of the service script to populate required elements before an embedded business object or business service is invoked from the service script.

```
<schema>
  <logDate mapField="LOG_DT" default="%CurrentDate" required="true" private="true"
</schema>
```

mdField="md field code"

The meta-data field element is used to associate an element with a field's metadata. Note that the field defines data type, as well as description, and (translatable) label. So, if you link an element with a meta-data field, you don't need to specify any of these attributes.

NOTE: If you are creating a business object schema, then the mapField attribute is equivalent to the mdField attribute. In other words, you don't need to specify both the mapField and mdField attributes unless you need to override other attributes like (translatable) labels, type, size, etc.

```
<schema>
  <active mdField="CM_ACTIVE_SW" />
</schema>
```

fkRef="fkRef"

You can link an element to a foreign key reference. This will enable framework validation of the element during schema interaction.

```
<schema>
  <person fkRef="PER" mapField="CHAR_VAL_FK1" >
    <row mapChild="CI_SA_CHAR" >
      <CHAR_TYPE_CD is="PER" />
    </row>
  </person>
</schema>
```

private="true"

Marking an element as private will prevent it from being exposed in schema interaction.

NOTE: A private element requires a default.

```
<schema>
  <type mdField="SA_TYPE_CD" default="E1" private="true"/>
</schema>
```

suppress="true | blank | input"

Marking an element as suppressed will prevent it from appearing in automatically generated user interface applications.

```
<schema>
```

```
<ls mdField="LIFE_SUPPORT_FLG" default="N" suppress="true"/>
</schema>
```

NOTE: The `suppress="true"` attribute can be specified on a group, in which case all elements of the group will be suppressed.

Marking an element as `suppress="blank"` means that the rendering engine will hide the element if its value is blank. The element will still be modifiable on the input map whether blank or not.

```
<schema>
  <email mdField="EMAIL" suppress="blank" />
</schema>
```

Marking an element as `suppress="input"` means that the rendering engine will suppress the element for input, although it may still be displayed if not blank.

NOTE: Elements marked as `suppress="input"` will behave as with `suppress="blank"`. If the value is blank, no value will be displayed on either the display or input map. If the element's value is present, then the element will be displayed on both the display and input map.

```
<schema>
  <email mdField="EMAIL" suppress="input" />
</schema>
```

noAudit="true"

Marking an element as not auditable will prevent it from ever appearing as a changed element in the business object's audit plug-in spot. If specified on a group or listnode it applies to the whole node. You cannot specify it on the schema root node, only on schema elements.

NOTE: The `noAudit` attribute is only applicable to business object schemas.

```
<schema>
  ...
  <version mapField="VERSION_NBR" noAudit="true" />
  ...
  <workFields type="group" noAudit="true"
    <lastProcessedId/>
    <lastProcessedTime/>
  </workFields>
</schema>
```

Schema Constants

There are some product owned schemas where the design warrants a value to be defaulted in the schema, but where the value is implementation specific and therefore cannot be defined by the product. For these situations, the product may use a technique called a schema constant. The design of the schema will include a declared constant. At implementation time, a configuration task will include defining the appropriate value for the constant.

For example, imagine the product delivers an algorithm that will create an outbound message when a certain condition occurs. The outbound message type to use must be configured by the implementation. To use a schema constant to define the outbound message type, the base product will configure the following:

- An option type lookup value for the lookup **F1CN_OPT_TYP_FLG** is defined. For example, **M202** — Activity Completion Outbound Message Type with a Java Value Name of **outmsgCompletion**
- The base schema that is used to create the “complete activity” outbound message references the schema constant using the Java Value Name of the option type's lookup value

```
...
<outboundMessageType mapField="OUTMSG_TYPE_CD" default="%Constant(outmsgCompletion)"/>
...
```


At implementation time, the administrative users must configure the appropriate outbound message type for “activity completion”. Then, navigate to [Feature Configuration](#), choose the **Schema Constants** feature type, choose the option type **Activity Completion Outbound Message Type** and enter the newly created outbound message type in the option value.

Schema constants may also be used in the flattening syntax to define [the row elements required for flattening](#).

Defaulting and System Variables

The default node can be used to default values into field elements as well as [the row elements required for flattening](#) . You can default a field to a constant or to one of several system variables.

NOTE:

When using the default attribute it is also necessary to specify the **required="true"** attribute, except when the default is used for the flattening syntax.

Mnemonic	Valid Values	Description	Examples
default=	"value"	Use this attribute to default an element to a specified value. The values that are valid depend on the dataType setting.	<pre><schema> <perType mapField="PER_OR_BUS_FLG" > </schema> <schema> <frequency dataType="number" defau > </schema></pre>
	"%CurrentDate"	Used to default the element to the current date. This is only applicable to date elements.	<pre><schema> <logDate mapField="LOG_DT" default > </schema></pre>
	"%CurrentDateTime"	Used to default the element to the current date / time. This is only applicable to date / time elements.	<pre><schema> <logDateTime mapField="LOG_DTTM" d > </schema></pre>
	"%StandardDateTime"	Used to default the standard date and time. The standard date and time is identical to the current date and time, unless daylight savings time / summer time is in effect for the base time zone. This may be used with the stdTime attribute. NOTE: Refer to Standard Time Considerations for more information.	<pre><schema> <startDateTime mapField="START_DTT > </schema></pre>
	"%ProcessDate"	You can default the process date. The process date differs from the current date because the process date will remain constant throughout the duration of the process being executed. The current date will reflect the actual date of processing. This is similar	<pre><schema> <billDate mapField="BILL_DT" defau > </schema></pre>

Mnemonic	Valid Values	Description	Examples
		to the batch business date that is a standard batch parameter .	
"%ProcessDateTime"		This is similar to "%ProcessDate" but for date / time fields.	<pre><schema> <calcDateTime mapField="CALC_DDTM" > </schema></pre>
"%CurrentUser"		Used to default the element to the current user.	<pre><schema> <logUser mapField="LOG_USER" defau > </schema></pre>
"%CurrentUserTimeZone"		Used to default the element to the current user's time zone. If the current user's time zone is not found, the installation time zone is used.	<pre><schema> <timeZone default="%CurrentUserTim > </schema></pre>
"%CurrentUserLanguage"		Used to default the element to the current user's language.	<pre><schema> <custLanguage mapField="CUST_LANG > </schema></pre>
"%InstallationCurrency"		Used to default the currency from the installation record.	<pre><schema> <currency mapField="CURRENCY_CODE > </schema></pre>
"%InstallationCountry"		You can default the country from installation record.	<pre><schema> <country mapField="COUNTRY" defaul > </schema></pre>
"%InstallationLanguage"		You can default the language from the installation record.	<pre><schema> <language mapField="LANGUAGE" defa > </schema></pre>
"%Constant(SchemaConstant)"		You can default an element value using a schema constant .	<p>The following is an example of a schema constant used as a default value, where the Java Value Name of the Lookup Value is 'customerLanguage'.</p> <pre><language mapField="CUSTOMER_LANG" c ></pre>
"%Context(ContextVariable)"		<p>You can default a value contained in a context variable.</p> <p>WARNING: Context variables must be initialized within a server script before the schema context default can be applied. Refer to Context Variables for more information.</p>	<p>An example of a context variable used as a default value:</p> <pre><source mapField="PER_ID" default="%" ></pre> <p>NOTE: When defining a context variable in scripting, it should be prefixed with \$\$. When referring to the variable in the %Context() syntax, the prefix is not included.</p>
defaultRef=	"XPath"	Use this attribute to default the value of one element to the value of another one.	Refer to Referencing Other Elements for supported syntax for referring to other elements.

The Flattening Nodes and Attributes

The term "flattening" is used to describe the act of defining one or more single elements for a schema that are actually part of a list within the maintenance object. Flattening is possible if there are other attributes of the list that can be defined to uniquely describe the element or elements. A common use case for flattening is a characteristic. Rather than defining the characteristics of an object using a collection where the user must choose the characteristic type and then define the value, the characteristics are defined as actual elements with the appropriate label already displayed. This technique enables the designer of the schema and the user interface to display each separate characteristic in the logical place in the user interface rather than all lumped together.

NOTE: A flattened element represents a unique row in the database. This row is inserted when the flattened values are created. The row is updated when any of the flattened values are changed. The row is deleted when all the flattened values are removed. The behavior of effective dated elements is slightly different - please see [Flattening an Effective Dated List](#).

NOTE: The flattening feature can also be used to define a list, see [Flattened List](#).

Identifying the List or Child Table

When flattening a child table, the row node is required to identify the list / child table that the element comes from. Within the row node, at least one element must be defined with an **is=** definition that ensures that the element is a unique row in the database. It may also define elements or fields in the row that are suppressed and are populated using default value configuration.

- For a business object, the row node defines the child table the flattened field belongs to.

The syntax is `<row mapChild="table name">`. This example is for the list of persons for an account in the customer care and billing product. One person may be marked as the "main" person. This illustrates how to define an explicit element for the main person ID to simplify references to that field. It is part of the `CI_ACCT_PER` child table. What makes it unique is that the `MAIN_CUST_SW` is **true** (and only one row may have that value)

```
<custId mapField="PER_ID" mdField="CM-MainCust">
<row mapChild="CI_ACCT_PER">
  <MAIN_CUST_SW is="true" />
  <ACCT_REL_TYPE_CD default="MAIN" />
</row>
</custId>
```

NOTE: The above example illustrates that the row node may also define elements within the list that are suppressed and assigned a default value. This syntax is never used to identify a particular row. Note that a default value can either be a literal string, or a [system variable](#).

- For a business service, the row node identifies the list name the flattened field belongs to.

The syntax is `<row mapList="list name">`. This example shows two entries from a list being flattened to a field value and description.

```
<selectList type="list" mapList="DE">
  <value mapField="COL_VALUE">
<row mapList="DE_VAL">
  <SEQNO is="2"/>
</row>
  </value>
  <description mapField="COL_VALUE">
<row mapList="DE_VAL">
  <SEQNO is="3"/>
</row>
</description>
</selectList>
```

Uniquely Identifying the Flattened Field or List

The `is=` syntax within a row or rowFilter element is used to uniquely identify the row.

Mnemonic	Valid Values	Description	Examples
<code>is=</code>	"value"	Use this attribute to reference a value directly.	<pre><tdTypeCd mapField="CHAR_VAL_FK1"> <row mapChild="F1_EXT_LOOKUP_VAL_C <CHAR_TYPE_CD is="CM- TD-TYPE" /> </row> </tdTypeCd></pre>
	"%Constant(SchemaConstant)"	You can configure a flattened element using a schema constant . During a service interaction the value of the schema constant will be used to identify the flattened element in its child row.	<p>An example of a schema constant used in flattening syntax where the Java Value Name of the Lookup field value is 'cmPrice'.</p> <pre><unitRate mapField="CHAR_VAL" dataTy <row mapChild="F1_EXT_LOOKUP_VAL_C <CHAR_TYPE is="%Constant(cmRate) > </row> </unitRate></pre>
	"%n"	The %n substitution value is used to reference a relative list instance. A relative list instance is typically used to configure a flattened element for a child table keyed by sequence number. The value of n should be a positive integer value. During a business object read interaction the relative list instance (position) specified by the integer will be returned.	<p>An example with a relative list instance - where the first instance of the row is returned.</p> <pre><schema> <mainPhone mapField="PHONE"> <row mapChild="CI_PER_PHONE"> <PHONE_TYPE_CD is="%Constant(n > <SEQ_NUM is="%1" / > </row> </mainPhone> </schema></pre>

FASTPATH: Additional values for `is=` are used when [Flattening an Effective Dated List](#). Refer to that section for more information.

Flattening a Pre-defined Characteristic Type

If the flattened field is in a characteristic collection and the characteristic type is a predefined characteristic, automatic UI rendering will generate a dropdown for the list of valid values. For example, the schema below will generate a dropdown for the Usage element showing the valid values of the Status Reason Usage (**F1-SRUSG**) characteristic type.

```
<usage mdField="STATUS_RSN_USAGE" mapField="CHAR_VAL">
  <row mapChild="F1_BUS_OBJ_STATUS_RSN_CHAR">
    <CHAR_TYPE_CD is="F1-SRUSG" />
    <SEQ_NUM is="1" />
  </row>
</usage>
```

Defining Multiple Elements from the List

When attempting to include multiple columns from the same list, the system provide shorthand notation for copying the flattening rules defined on another element so that the flattening rules do not need to be repeated. To do this, the row node includes the **rowRef** attribute and it indicates the other element name that defines the mapping information. The

following example illustrates flattening the fields Customer ID and Receives Copy of Bill from the same list of Persons for an Account (where the MAIN_CUST_SW is **true**).

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" />
    <ACCT_REL_TYPE_CD default="MAIN" />
  </row>
</custId>
<copyBill mapField="RECEIVE_COPY_SW" rowRef="custId"/>
```

Note that the above notation also illustrates that the **rowRef** may be defined directly in the element's attribute definition.

NOTE: Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

Flattening Two Layers Deep

If your maintenance object or service has nested lists two layers deep, the system supports flattening and element within a flattened element. This technique also uses the **rowRef** attribute. The flattening of the second level refers to the flattened element of the first level. The following example illustrates flattening a characteristic into an element called legalContact for the “main” customer. Notice that the legalContact element has two row nodes: one to refer to the flattening information for its parent record and one to define its child table

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" />
    <ACCT_REL_TYPE_CD default="MAIN" />
  </row>
</custId>
<legalContact mapField="CHAR_VAL_FK1">
<row rowRef="custId">
  <row mapChild="CI_ACCT_PER_CHAR" >
    <CHAR_TYPE_CD is="LEGAL" />
  </row>
</row>
</legalContact>
```

Note that the above notation also illustrates that the **rowRef** may be defined as an attribute of a row node rather than directly in the element's attribute definition.

Defining a Flattened List

There are times that a list or child table supports multiple values of the same “type” and these should be presented as a list. To continue with the example above, the list of persons for an account may identify one person as the “main” person. This person has been flattened to a single element (with the account relationship type defaulted and suppressed). To maintain the other persons related to an account, you can define a list where each row captures the Person Id and the Account Relationship Type.

Rather than a row node, the flattened list is configured with a **rowFilter** element. The following schema illustrates the described example. The list node defines the child table. The **rowFilter** includes the criteria that identify the rows within the table to include. The elements of the list are defined within the list node outside the **rowFilter** element.

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" />
    <ACCT_REL_TYPE_CD default="MAIN" />
  </row>
</custId>
<miscPersons type="list" mapChild="CI_ACCT_PER">
  <rowFilter>
    <MAIN_CUST_SW is="false" />
  </rowFilter>
  <relType mapField="ACCT_REL_TYPE_CD"/>
  <personId mapField="PER_ID"/>
</custId>
```

Note that the system will validate that if a schema contains flattened single elements and flattened lists from the same child table, the criteria that defines what makes them unique must be analogous.

Flattening an Effective Dated List

There are some lists in the application that are effective dated (and still others that have effective date and time). For example, there are some effective dated characteristic collections. In these collection, the design is to capture a single value for a characteristic type that may change over time. It is not meant to support multiple characteristic values in effect at the same time. The following highlights some information regarding effective dated characteristic functionality:

- The most recent dated row is returned when invoking a BO for read.
- No new row added when all of the values are unchanged on a change to the BO.
- The flattened row value is updated when any of the flattened values are changed and the most recent date is equal to the current date (or the referenced effective date);
- A new row value is inserted when any of the flattened values are changed and the most recent date is different than the current date (or the referenced effective date);

NOTE: Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

When flattening an effective dated list, the date column must include information regarding the date to use. The following table highlights the possible values.

Mnemonic	Valid Values	Description	Examples
is=	"%effectiveDate""	Use this configuration to indicate that current date should be used for processing. Any value added or updated using this schema will be for the current date. With this option, if the maintenance object allows for the characteristic value to be blank, then setting the flattened value to blank during the BO update will result in updating the existing record with an empty value, or adding a new row with an empty value in case the current date effective dated record is not found.	<pre><schema> <price mapField="CHAR_VAL" dataTyp <row mapChild="CI_SA_CHAR"> <CHAR_TYPE is="PRICE"/ > <EFFDTis="%effectiveDate"/ > </row> </price> </schema></pre>
	"%effectiveDate(reference element)"	Use this configuration to indicate that the date to use the value of another element. NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.	<pre><schema> <price mapField="CHAR_VAL" dataTyp <row mapChild="CI_SA_CHAR"> <CHAR_TYPE is="PRICE"/ > <EFFDTis="%effectiveDate(price </row> </price> <priceEdate mapField="EFFDT" rowRef= > </schema></pre>
	"%effectiveDateTime"	Use this configuration to indicate that current date /time should be used for processing. Any value added or updated using this	<pre><schema> <price mapField="CHAR_VAL" dataTyp <row mapChild="RATE_CHAR"> <CHAR_TYPE is="PRICE"/ ></pre>

Mnemonic	Valid Values	Description	Examples
		schema will be for the current date / time.	<pre><EFFDTis="%effectiveDateTime" > </row> </price> </schema></pre>
	"%effectiveDateTime(reference element)"	<p>Use this configuration to indicate that the date / time to use the value of another element.</p> <p>NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.</p>	<pre><schema> <price mapField="CHAR_VAL" dataTyp <row mapChild="RATE_CHAR"> <CHAR_TYPE is="PRICE"/ > <EFFDTTmis="%effectiveDateTime > </row> </price> <priceEdatetime mapField="EFFDTTM > </schema></pre>

Search Zone

A UI Map schema element can be configured to enable an automatic search dialog when the schema is included within a maintenance UI map.

NOTE: Please note that an fkRef can be configured with a search zone. If a schema element has an fkRef but no explicit search attributes (as described here) then the fkRef search information will be used in the UI map. In other words, if the schema element already has an fkRef, then these explicit search attributes in the schema are only used to override the fkRef search information.

NOTE: Refer to the [UI Map Attributes and Functions](#) for more information on search zone configuration.

search="search zone"

The search attribute can be used within a UI map schema and is used to automatically generate the oraSearch UI map attribute. The search zone is an explorer zone configured as a search.

```
<person fkRef="PER" search="C1_PSRCH" />
```

searchField="search field:element['literal'];"

The searchField attribute can only be used in conjunction with the search attribute. The searchField attribute is used to build the oraSearchField UI map attribute. The searchField value is used to populate a search zone filter with an initial value when the search zone is launched. The initial value can be a literal also. The searchField value is used to match to the filter mnemonic also named searchField.

Search field: element['literal']. The search field represents the search zone filter to populate on launch. The element is the map's schema element used to populate the filter. The element is optional, if left blank, it will default to the element that this attribute is specified on. The searchField also takes 'literal' as input value

NOTE: Multiple filters can be populated within the search zone at launch, but multiple search field pairs must be constructed within the attribute value. The value specified here will be used to directly build the HTML attribute oraSearchField within the UI map where this schema is specified.

NOTE: Note that the element reference is *relative*. Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

```
<person fkRef="PER" search="C1_PSRCH" searchField="PERSON; PER_TYPE:personType;" />
```

searchOut="search field:element;"

The searchOut attribute can only be used in conjunction with the search attribute. The searchOut attribute is used to build the oraSearchOut UI map attribute. The searchOut value is used to capture a selected value from the search zone and move it to a UI map element. The searchOut value specified should match the ELEMENT_NAME mnemonic within the search result zone parameter.

Search field: element. The search field represents the search zone result brought back to the UI map. The element is the map's schema element to be populated. The element is optional, if left blank, it will default to the element that this attribute is specified on.

NOTE: Multiple elements can be populated as a result of search zone selection, but multiple search field pairs must be constructed within the attribute value. The value specified here will be used to directly build the HTML attribute oraSearchOut within the UI map where this schema is specified.

NOTE: Note that the element reference is *relative*. Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

```
<person fkRef="PER" search="C1_PSRCH" searchField="PER_ID" searchOut="PER_ID;PRIMARY_PHONE:personPhone;" />
```

Extend Security for Service Script

Application service security will be enforced when either a business object or a service script is invoked from a BPA script or a UI map, but not from a service script. If you want security to be enforced when the business object or a service script is invoked from a service script, you must add the following attribute to the service script's schema.

appSecurity="true"

The appSecurity attribute is only available for service script schemas. If specified, any business object or service script directly invoked by the service script will have their application service evaluated for access.

```
<schema appSecurity="true">
  ...
</schema>
```

Overriding Action for a Business Service

If you want to invoke a business service with an action other than 'read', you need to specify the action attribute on the primary node business service schema.

pageAction="add, change, delete"

The action attribute is used to override the default action of read on a business service schema. Valid values are:

- add
- update (only allowed for maintenance object service)
- change (not allowed for maintenance object service)
- delete
- read (this is the default action if no pageAction specified)

Example:

```
<schema pageAction="change">
  <parm type="group">
    <ele1/>
    <ele2/>
  </parm>
</schema>
```


Specifying searchBy for a Search Service

If you want to invoke a search service then you must explicitly specify the searchBy attribute appropriate for the elements mapped in the schema.

searchBy="MAIN"

The value values of the searchBy attribute can be found by viewing the XML schema linked to the business service, use the View XML url. Typical values are:

- MAIN
- ALT
- ALT2
- ALT3 (etc.)

```
<schema searchBy="MAIN">
  <AccountID mapField="ACCT_ID" />
  <Results type="list">
    <AccountID mapField="ACCT_ID" />
  </Results>
</schema>
```

Including Other Schemas

There are no limitations on your ability to include a schema into another schema - all types can be included in all other types. Nested includes are also allowed - and at present there is no limitation on the depth of the nesting.

Including a schema requires two parts:

1. The include node
2. The name attribute

<includeBO name="schema name"/>

Including a business object schema into another schema is allowed. However, note that the mapping rules of a business object or business service schema may or may not make sense in the context of the parent schema. Include other schemas at your own risk. However, a very useful aspect of XML processing is that the framework ignores non-pertinent attributes. In other words, it will not hurt to have mapping attributes included into a script schema.

```
<schema>
  <cust type="group">
    <includeBO name="Person" />
  </cust>
  <spouse type="group">
    <includeBO name="Person" />
  </spouse>
</schema>
```

<includeBS name="schema name"/>

Include business service schema.

```
<schema>
<includeBS name="RateApp" />
</schema>
```

<includeDA name="schema name"/>

Include stand alone data area schema.

```
<schema>
<includeDA name="Source Info" />
</schema>
```

<includeMP name="schema name"/>

Include user interface map schema.

```
<schema>
<includeMP name="Customer" />
</schema>
```

<includeSS name="schema name"/>

Include service script schema.

```
<schema>
<includeSS name="NewCustomer" />
</schema>
```

Compatibility Attributes

These attributes were added as part of upgrades from previous versions of the Framework.

fwRel="2"

This attribute has been added to schemas created in Framework 2 as part of a Framework 4 upgrade. New schemas will not need this attribute. It is not advisable to modify this attribute without understanding the following behavior differences as improper changes could result in errors:

NOTE: Schemas created in Framework 2 with the fwRel="2" attribute will store any XML mapped fields under groups as top-level XML elements in the mapXML field. This means that if two or more fields, in different group structures, were to have the same field name, their storage would conflict with one another resulting in errors. The new behavior, without the fwRel="2" attribute, will preserve the group structure and avoid the conflicts.

```
<schema fwRel="2"
...
</schema>
```

UI Hint Syntax

Contents

- [Working Examples](#)
- [Technical Notes](#)
- [Format an Input Map Title](#)
- [Create a Section](#)
- [Include a Map Fragment](#)
- [Build Dropdown](#)
- [Conditionally Hide Elements](#)
- [Conditionally Protect Elements](#)
- [Trigger Dependent Behavior](#)
- [Control Rendering Target](#)
- [Generate a Text Area](#)
- [Modify FK Reference Defaults](#)
- [Suppress Automatic Number Formatting](#)
- [Auto Capitalize the Input Data](#)

Working Examples

For working examples of uiHint functionality, refer to the following business objects:

BOs with User Assigned Keys

The following examples illustrate the patterns used to enable uiHints on an object with a user specified key.

- **F1-OutcomeStyleLookup.** This extendable lookup BO does not require state transition, but does allow duplicate and delete actions.
- **F1-TodoSumEmailTyp.** This request type illustrates the hints required to support state transition on a display map.
- **F1-WebSvc.** This web service BO is a good example for management of complex JavaScript requirements. Both display an input maps have functionality that require specialized javascript.

BO with System Generated Key

The following example illustrates the pattern used to enable uiHints on an object with a system generated key.

- **F1-GenericAttachment.** This attachment BO has a system assigned key, which entails the following special handling:
 - **Main Section Data Area: F1-AttachmentMain.** This data area contains the elements common to all attachments, including the key, bo, and version. Because this data area is used to define the main section of the generated maps, the main section of the map can be extended by an implementation via data area extension functionality.
 - **Record Actions Fragment: F1-AttachmentActions.** This map contains the standard actions, Edit and Delete, plus custom actions used only by attachments, View and Upload.
 - **Record Information Fragment: F1-AttachmentIDFrag.** This map contains the primary key of the attachment.

Display Map Service Script

Display map service scripts can be fully supported via dynamic HTML generation. However, to help eliminate the need for a display service script, self-contained uiHint functionality has been developed to write the business object status and determine valid state transitions. So the two most common reasons to craft a display service script have been eliminated.

A typical reason to use a display pre-script is if you have an embedded map fragment that contains a business service schema. The display service script can be used to invoke the business service. Both the map fragment and the display service script must declare the business service schema to support this scenario.

WARNING: The zone used to display the object's map must have a derivation script, like **F1-GncDsMpDZ** or **F1-GenDss**, that will invoke a display service script for the business object if it has been defined as a BO option - but not require an explicit display map BO option. In addition, the display service script's schema must be enabled for uiHint functionality - as the script's schema will be dynamically rendered by the zone - and not the BO schema.

- **F1-ExcelSpreadsheet.** This attachment BO has a display service script used to manipulate the attachment business object before displaying it:
- **Display Service Script: F1-AtchDtlU.** This service script's schema has been defined with the uiHint namespace, and will have a display map generated for it.

Maintenance Pre-Processing Service Script

Maintenance pre-processing service scripts can be used with uiHints.

- **F1-ExcelSpreadsheet.** This attachment BO has a maintenance pre-processing service script used to manipulate the attachment business object before rendering the maintenance map:
- **Pre-Processing Service Script: F1-AtchPre.** This service script's schema mimics a maintenance map schema with embedded boGroup and action elements. It will be invoked before the maintenance map is rendered.

Maintenance Post-Processing Service Script

Maintenance pre-processing service scripts can be used with uiHints.

- **F1-ExcelSpreadsheet.** This attachment BO has a maintenance post-processing service script used to manipulate the attachment business object after rendering the maintenance map:
- **Post-Processing Service Script: F1-AttchPost.** This service script's schema mimics a maintenance map schema with embedded boGroup and action elements. It will be invoked after the maintenance map is rendered.

Technical Notes

The following prerequisites are required to support dynamic html generation:

Schema Requirements

To support automated UI generation, the business object schema must contain the following:

- `<schema xmlns:uiHint="http://oracle.com/ouafUIHints">`. The schema node must name the uiHint namespace.
- `isPrimeKey="true"`. Every element of the business object schema that is part of the primary key must be identified.

Maintenance Script Requirements

The maintenance script for the MO must be enabled for dynamic generation.

CAUTION: The business object maintenance BPA script must be declared as an MO Option for uiHint maintenance functionality to work!

If the script performs **F1-BOProc** then it is likely no special functionality is needed. However, if the maintenance script contains its own call to **F1-GetValOpt** then the following statement is required prior to that call:

```
move 'false' to "F1-GetBOOpts/input/maintenanceMapRequired";
performScript 'F1-GetValOpt';
```

After the call to **F1-GetValOpt** the following logic must be included to dynamically declare the map schema if the business object does not have a maintenance map of its own:

```
// Perform Main Processing
if ("F1-GetBOOpts/output/maintenanceMap = $BLANK")
  declareBOWithBOGroup "$bo" as 'map_schema';
else
  declareMap "F1-GetBOOpts/output/maintenanceMap" as 'map_schema';
end-if;
```

Format an Input Map Title

A uiHint element can be used to build a title for a maintenance map. The title will only print on the maintenance map, not on the display map. It will be printed as the first line in the map, centered, with a heading style.

The syntax of this element is `<uiHint:title mdField="field name" label="text"/>`

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
<uiHint:title mdField="STATUS_RSN_LBL"/>
  ...
</schema>
```

Create a Section

The uiHint namespace supports the definition of a UI map section. Note that sections are currently created in generated UI Maps when the schema has a group or list node with a label or mdField. The functionality described here enables the creation of a section without requiring a labeled group or list node within the schema. Every section must be bounded by startSection and endSection element pair.

The syntax for the start section attribute is `<uiHint:startSection sectionColumn="left | right | fullWidth" sectionOpen="false" mdField="field name" label="text" visibleOn="displayMap | inputMap"/>`

The supporting attributes for the section are as follows:

- **sectionColumn**. Specify this attribute if you want the section to appear in either the left or right column. If you do not specify this attribute then the section will default to full width.
- **sectionOpen**. Specify this attribute if you want the section to initially display as a closed section. If you do not specify this attribute, the section will default as open on initial display.
- **mdField** or **label**. Specify either the name of a field the contains the section's label or define the label directly.
- **visibleOn**. Specify this attribute when the section is not applicable to both the display and the input generated UI maps. Indicate which map it should be visible on and it is suppressed on the other map.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <uiHint:startSection label="Main" sectionColumn="left"/>
  ...
</schema>
```

NOTE: The sectionColumn and sectionOpen attributes are available for group and list nodes as well.

The syntax for the end section attribute is **<uiHint:endSection/>**

Every section must be bounded by a startSection and endSection pair.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <uiHint:startSection label="Main" sectionColumn="left"/>
  <includeDA name="F1-ExtLookupCommon"
  <uiHint:endSection/>
  ...
</schema>
```

Include a Map Fragment

You can specify a UI map fragment to inject HTML into a generated map using the **includeMap** element name. This allows for you to support more sophisticated behavior on your user interface. For any element that is included for rendering in the map fragment, be sure to suppress the element in its schema definition, otherwise HTML will automatically be generated for the element.

The syntax is **<uiHint:includeMap map= "map name" visibleOn="displayMap|inputMap"/>**. Besides specifying the map name, you may indicate whether the map should be shown in the display map or the input map.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <uiHint:includeMap map="StandardActionButtons" visibleOn="displayMap"/>
  ...
</schema>
```

NOTE: Important note on the map fragment schema: If a map fragment contains a schema, then the fragment schema structure will be injected into the dynamically generated schema when the business object is rendered for input. Technically, the fragment schema will be inserted after the boGroup structure within the map's schema. This method may be used to support the implementation of maintenance pre and post script processing for a business object and oraInvokeBS function calls within embedded JavaScript.

If JavaScript is required within an XHTML UI Map fragment, it is necessary to bound it within a **![CDATA[]]** tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
/*  */
//
//javascript
//
/* ]&gt; */
&lt;/script&gt;</pre>
</div>
<div data-bbox="88 853 890 883" data-label="Text">
<p><b>Flush the cache:</b> For performance reasons, the Framework automatically caches business object schemas, data areas, and UI maps. When you update a business object, the cache is automatically flushed. However, if the business object includes</p>
</div>
<div data-bbox="418 944 907 959" data-label="Page-Footer">
<p>Oracle Utilities Customer Care and Billing Administrative User Guide • 253</p>
</div>
```

either a data area or embedded UI map fragment, the cache must be manually flushed in order for your changes to be recognized. Refer to [Server Cache](#) for more information.

Build A Dropdown

Syntax is provided to build a dropdown list in an edit map. The dropdown may be built using data returned from a service script, a business service or a table.

The syntax is **uiHint:select="ss:SS Name | bs:BS Name | table:Table Name" selectIn="servicePath:element | 'literal';" selectOut="valuePath:servicePath; descPath:servicePath"**

When specifying a table, only the table name must be defined. When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.

- **selectIn** is used to pass input data into the service. For every input parameter, specify the XPath reference to the element in the service's schema followed by either a literal value or the XPath reference to the element whose value should be passed in.
- **selectOut** is used to specify the elements in the service's schema that identify the returned values and descriptions.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <boStatus mapField="BO_STATUS_CD" uiHint:select="bs:F1-
BOStateReasonList" uiHint:selectIn="boStatusBO:boStatusBO" uiHint:selectOut="valuePath:results/
status; descPath:results/description"/>
  ...
  <algorithm mdField="ALG_CD" uiHint:select="bs:F1-
RetrieveSysEvtAlgorithms" uiHint:selectIn="algorithmEntity:'F1AA';" uiHint:selectOut="valuePath:results/
algorithm;descPath:results/description"/>
  ...
  <outboundMsgType mdField="OUTMSG_TYPE_CD" required="true" uiHint:select="table:F1_OUTMSG_TYPE" fkRef="F1-
OMTYP"/>
</schema>
```

Conditionally Hide Elements

The **displayNone** attribute is used to suppress elements on the map based on conditions.

The syntax is **uiHint:displayNone="'xpath'|function,'value','!='|'='"**

The following points describe the possible configuration.:

- *'xpath'* identifies an element via XPath whose value should be interrogated. When using this option, enter the value to compare using *'value'*. A blank value is specified with ''.
- *function* is a JavaScript function, which must return a Boolean. When using this option, the *'value'* used in this expression must be **true** or **false**.
- For any of the above settings, you may optionally use an operator of **!='** or **'=**. By default, the operator is **'=**.

WARNING:

Embedded spaces are not supported within the comma separated string values of the uiHint:displayNone function.

The displayNone uiHint may be used on group nodes, list nodes, and elements - except for elements within a list. Elements within a list cannot be hidden conditionally.

The following example illustrates that the currency reference element will be hidden when the data type element is not equal to 'money' (**F1MO**). Note that this example also illustrates [Trigger Dependent Behavior](#) because the data type element's value may change and if it does, the condition for hiding the currency reference should be re-evaluated.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <dataType mdField="F1_SE_DATA_TYPE" dataType="lookup" lookup="F1_SE_DATA_TYPE" uiHint:dependents="currenc
>
  <currencyRef mdField="F1_SE_CURR_REF_LBL" uiHint:displayNone="'dataType','F1MO','!='/'>
  <lookup mdField="F1_SE_LOOKUP_LBL" fkRef="F1-LKUPF" uiHint:displayNone="'dataType','F1LP','!
='/'>
```

```
...
</schema>
```

The following example illustrates referring to a function where the function receives parameters:

```
<uiHint:startSection mdField="F1_SE_DEFAULT_SECT" uiHint:displayNone="isApplicableForSchemaType(item, 'F1MP'),
>
```

Conditionally Protect Elements

The protect attribute is used to protect elements on the map based on other factors.

The syntax is **uiHint:protect**="'*xpath*'|*function*|*action*',*value*',*!=*|*='*'"

The following points describe the possible configuration.:

- '*xpath*' identifies an element via XPath whose value should be interrogated. When using this option, enter the value to compare using '*value*'. A blank value is specified with ''.
- *function* is a JavaScript function, which must return a Boolean. When using this option, the '*value*' used in this expression must be **true** or **false**.
- '**action**' indicates that the protection is based on the current action. For example, certain fields may only be specified when adding a record. Any subsequent changes to the record should protect the field from being changed. When using this option, the valid values for the '*value*' are **A** (add) and **C** (change).
- For any of the above settings, you may optionally use an operator of '*!=*' or '*='*'. By default, the operator is '*='*'.

CAUTION:

Embedded spaces are not supported within the comma separated string values of the **uiHint:protect** function.

The protect UI Hint may be used on group nodes, list nodes, and elements - except for elements within a list. Elements within a list cannot be protected conditionally.

The following UI Hint will protect the currencyRef element when the dataType element is not equal to 'money':

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
...
<dataType mdField="F1_SE_DATA_TYPE" dataType="lookup" lookup="F1_SE_DATA_TYPE" uiHint:depends="currencyRe
>
<currencyRef mdField="F1_SE_CURR_REF_LBL" uiHint:protect="'dataType', 'F1MO', '!='"/>
<lookup mdField="F1_SE_LOOKUP_LBL" fkRef="F1-LKUPF" uiHint:protect="'dataType', 'F1LP', '!='"/>
...
</schema>
```

Trigger Dependent Behavior

The depends attribute is used to trigger behavior on a child element when a parent element is changed.

The syntax is **uiHint:depends**="'*element*,'" where a list of dependent elements may be indicated. Use a semi-colon to delimit multiple dependent element names.

The following example illustrates that the dropdown list of one element is driven by the value of another element. In this example, when the Country changes, the list of States to choose from should change to only show the states for the indicated country.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <country label="Country" uiHint:select="table:CI_COUNTRY" uiHint:depends="state;"/>
  <state label="State" uiHint:select="ss:CM-RetrieveCountryStates" uiHint:selectIn="input/
country:country;" uiHint:selectOut="valuePath:output/state/stateCode; descPath:output/state/
stateDesc"/>
...
</schema>
```

NOTE:

Dependent targets may only name elements, not group or list nodes.

Do not modify the "id" attribute value of dependent and parent element. Data population in dependent is done based on the "id" attribute value.

Control Rendering Target

By default all elements that are not suppressed are visible on both the display map and the input map. Use the **visibleOn** attribute to limit the inclusion of an element to either the display or input map.

The syntax is **uiHint:visibleOn="displayMap | inputMap"**

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <uiHint:includeMap map="StandardActionButtons" visibleOn="displayMap"
  ...
</schema>
```

Generate a Text Area

By default, a standard text box is rendered in an input map for any string element. If the field is larger and you wish to have a bigger text area (with a scroll bar), use the **textArea** attribute.

The syntax is **uiHint:textArea="true"**

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <message label="Message" uiHint:textArea="true"/>
  ...
</schema>
```

Modify FK Reference Defaults

By default, when an element with **fkRef** is displayed, an info string, context menu, navigation, and search are enabled (if the FK reference has been configured accordingly). Syntax is provided to allow you to selectively turn off any of these features.

The syntax is **uiHint:fkRef="info:false;context:false;navigation:false;search:false;"**. Only the feature that you wish to turn off needs to be specified. The following example illustrates turning off the navigation capability, meaning the text will not be rendered as hypertext.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <attachmentID fkRef="F1-
ATTCH" primeKey="true" suppress="input" uiHint:fkRef="navigation:false;"/>
  ...
</schema>
```

Suppress Automatic Number Formatting

The **alphaFormat** attribute is used to turn-off the automatic formatting on numeric fields (**dataType="number"**). By default, its value is **false**, which means that the default numeric formatting will occur.

Note: If **dataType** is not specified explicitly, it is derived from **mdField** or **mapField**.

The syntax is **uiHint:alphaFormat="true|false"**

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <numberCount mdField="" dataType="number" uiHint:alphaFormat="true"/> <!--
numberCount is numeric field-->
  <numberCount mapField="" uiHint:alphaFormat="false"/> <!--
No effect on formatting. Attribute won't appear in the resultant xhtml -->
  <numberCount dataType="number" uiHint:alphaFormat="false"/> <!--
No effect on formatting. Attribute won't appear in the resultant xhtml -->
  ...
</schema>
```


Auto Capitalize the Input Data

The `uiHint` provides syntax to automatically capitalize input data.

Note: The attribute is only available in the schema designer when the `isPrimeKey` is set to `true`. The attribute may be added to any string element when using the source viewer.

The syntax is `uiHint:capitalize="true|false"`. By default, its value is `false` (and therefore can be left out altogether).

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <toDoTypeCd mdField="TD_TYPE_CD" uiHint:capitalize='true' isPrimeKey="true"/>
</schema>
```

NOTE: This field is ignored if `uiHint:textArea="true"` is configured.

Schema Designer

The Schema Designer is a user-friendly interface for performing the following common schema editing tasks:

- Displaying existing schemas.
- Creating schema elements.
- Moving elements within a schema.
- Adding attribute values.

The designer provides the ability to toggle between **Design View** and **Source View** views by clicking the icons on the **Schema Designer** title bar:



The **Source View** shows the schema elements and their attributes written in the proper syntax.

In the graphical **Design View** mode, the zone is split into two areas. The left pane displays the elements of the schema in a tree-like presentation, while the right pane displays all valid attributes and values for selected schema elements.

The following points highlight some functionality available in the design view.


- If the schema definition refers to another schema using an “include” statement, one can expand that schema in the left panel to view the elements within that schema. Clicking an element in that expanded view shows information about the element’s definition in the right panel. Changes cannot be made to elements in this view.
- To move an existing element within the schema in the left panel, simply drag and drop.
- Right clicking an element in the left panel causes a pop-up to display which offers options based on the position of the element clicked on and based on the type of element.
 - **Add element above | below**
 - **Delete this element**
 - **Move this element up | down**
 - **Add child element**

When adding a new element, you are prompted for the element type. The following lists the possible element types. Most are self explanatory and represent standard schema options.

- **Characteristic.** This is a special type of Flattened Field element that is used to map a single element to the characteristic for a given characteristic type. This element is only applicable for maintenance objects that have one or more characteristic child tables where the primary key is the maintenance object’s key, characteristic type and sequence.

Effective dated characteristic collections are not supported. The user defines the element name and the characteristic type. The system will configure the remaining flattening information accordingly.

- **Characteristic List.** This is a special type of Flattened List element that is used to map list element that includes a sequence and a characteristic value for a given characteristic type. This element is only applicable for maintenance objects that have one or more characteristic child tables where the primary key is the maintenance object's key, characteristic type and sequence. Effective dated characteristic collections are not supported. The user defines the list name, the characteristic value element name and the characteristic type. The system will configure the remaining flattening information accordingly.
- **Comment.** This adds a comment to the schema.
- **Embedded HTML.** This is specific to a schema enabled for UI Hints. It is used to include a UI map fragment.
- **Field**
- **Flattened Field**
- **Flattened List**
- **Group**
- **Include BO Schema**
- **Include BS Schema**
- **Include DA Schema**
- **Include Map Schema**
- **Include SS Schema**
- **Input Map Title.** This is specific to a schema enabled for UI Hints. It is used to define a title element for the map.
- **List**
- **Nested Flattened Field.** This is a flattened field from a child table.
- **Raw Element.** This element is used to capture text as is. It is typically used to capture an XML structure without any details of the definition of the individual nodes.
- **Section.** This is specific to a schema enabled for UI Hints. It is used to define a section within the map.
- **Simple Field.** This is a special type of Field element that is used to define an element that is mapped to a column that supports data defined in an XML structure. (This is either a column with the character large object data type (CLOB) or the XML data type).

Context-sensitive embedded help is provided for fields and controls in the right pane by clicking the Help icon 

The **Schema Designer** is available by choosing the **Schema** tab on the [Business Object](#), [Data Area](#), [UI Map](#), [Business Service](#), and [Script](#) pages.

Schema Viewer

The schema viewer shows a tree-view presentation of a schema in its expanded form.

The schema illustrates the structure to be used when communicating with the schema's associated object. The following takes place when a schema is expanded:

- If the schema definition includes references to other schemas, these references are replaced with the corresponding schema definitions.
- Also, if the schema definition contains **private** elements, they are omitted from this view.

Clicking on any node on the tree populates the text box on the top with the node's absolute XPath expression. You will find this feature very useful when writing scripts interacting with schema-based objects. [Scripting](#) often involves referencing

elements in a schema-based XML document using their absolute XPath expression. You can use this feature on the schema viewer to obtain the XPath expression for an element and copy it over to your script.

Business Event Log

Business Event Log may be viewed as a tool designed to capture any type of business event worth noting. You configure business objects to represent the various types of events your application calls for. The following type of details may be captured for each event:

- The business object representing the type of event.
- The date and time the event took place and who initiated it.
- The business entity for which this event is logged.
- Standard application message to describe the event.
- Additional context information that is available at the time of the event and varies for each type of event. The Business Event Log maintenance object supports a standard characteristics collection as well as an XML storage (CLOB) field. The event's business object determines where each piece of information resides. Refer to [Business Objects](#) for more information.

One common type of event may be the audit of changes made to sensitive data, for example, tracking an address change. Whenever an entity associated with a business object is added, changed, or deleted the system summarizes the list of changes that took place in that transaction and hands them over to **Audit** business object algorithms to process. You may design such an algorithm to audit the changes as business event logs. Refer to [a business object may define business rules](#) for more information.

You can also allow users to initiate business event logs to capture important notes about a business entity by exposing a [BPA Script](#) to invoke the event's corresponding business object.

Bottom line is that any process can create a business event log by invoking the business object representing the appropriate type of event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI_BUS_EVT_LOG](#).

Miscellaneous Topics

The following sections describe miscellaneous system wide topics.

Module Configuration

The system provides the ability to simplify the user interface based on functionality areas practiced by your organization.

Menu items and other user interface elements are associated with function modules. By default, all function modules are accessible. If a function module is not applicable to your business you may turn it off. Refer to [Turn Off A Function Module](#) for more information on how to turn off a module.

If a function module is made non-accessible, i.e. turned off, its related elements are suppressed from the user interface. In addition the system may validate that related functionality is not accessed. This also means that turning off the wrong module may cause any of the following to occur:

- Menu items may not appear. Refer to [Menu Item Suppression](#) to better understand how menu item suppression works.
- Entire menus may not appear. Refer to [Menu Suppression](#) to better understand how menu suppression works.
- Tabs on pages may not appear.
- Fields may not appear.

- The system may return an error message when you attempt to use a function (indicating the function is turned off).

To correct the above situation, simply remove the module from the turned off list thus making it accessible again.

Your module configuration setup is displayed on the [installations](#) record.

Menu Item Suppression

The following points describe how your module configuration can suppress [menu items](#).

- Menu items that are owned by the base product (as opposed to those your implementation adds) are associated with one or more function modules. If your module configuration has turned off all of the menu item's modules, the menu item is suppressed. If at least one of the modules is accessible, i.e. turned on, the menu item is not suppressed.
- If a menu line doesn't contain any accessible items, the menu line is suppressed.
- If all lines on a menu are suppressed, the menu itself (**Menu** or **Admin menu**) is suppressed in the application toolbar.

Menu Suppression

In addition to the above Menu Item Suppression logic, the following points describe how your module configuration can suppress an entire menu.

- Menus that are owned by the base product (as opposed to those your implementation adds) are associated with one or more function modules.
- If your module configuration has turned off all of the menu's modules, the entire menu is suppressed. If at least one of the modules is accessible, i.e. turned on, the menu is not suppressed.

Turn Off A Function Module

The base package is provided with a **Module Configuration** [Feature Configuration](#) that allows your organization to turn off base package function modules.

To turn off any of the base package function modules add a **Turned Off** option to this feature configuration referencing that module. Refer to the **MODULE_FLG** lookup field for the complete list of the application's function modules.

Any module not referenced on this feature configuration is considered turned on, i.e. accessible. To turn on a module, simply remove its corresponding **Turned Off** option from this feature configuration.

You may view your module configuration setup on the [installation options](#) page.

NOTE: Only one. The system expects only one **Module Configuration** feature configuration to be defined.

Global Context Overview

The framework web application provides each product the ability to nominate certain fields to act as a "global context" within the web application. For example, in Oracle Utilities Customer Care and Billing, the global context fields include Account ID, Person ID and Premise ID. The values of these fields may be populated as a result of searching or displaying objects that use these fields in their keys. If you navigate to the Bill page and display a bill, the global context is refreshed with the Account ID associated with that bill. The global context for Person ID and Premise ID are refreshed with data associated with that account.

The fields designated as global context for the product are defined using the lookup **F1_UI_CTXT_FLDS_FLG**.

Changing the values of the global context typically cause data displayed in zones on the dashboard to be refreshed to show information relevant to the current values of these global context fields.

When the value of one of the global context fields changes, an algorithm plugged into the [installation record](#) is responsible for populating the remaining global context values accordingly. Refer to your specific product for more information about the base algorithm that is provided for that product.

System Data Naming Convention

There are several maintenance objects in the system that include owner flag in one or more of its tables. We refer to the data in these tables as "system data". Some examples of system data tables include Algorithm Type, Batch Control, Business Object and Script. Implementations may introduce records to the same tables. The owner flag for records created by an implementation is set to **CM** (for customer modification), however the owner flag is not part of the primary key for any of the system data tables. As a result, the base product provides the following guidelines for defining the primary key in system data tables to avoid any naming conflict.

Base Product System Data

For any table that includes the owner flag, the base product will follow a naming convention for any new data that is owned by the base product. The primary key for records introduced by the product is prefixed with **xn-** where **xn** is the value of the owner flag. For example, if a new background process is introduced to the framework product, the batch code name is prefixed with **F1-**.

NOTE: There are some cases where the hyphen is not included. For example, portal codes omit the hyphen.

For most system data, the remainder of the primary key is all in capital case. An exception is schema oriented records. For business objects, business services, scripts, data areas and UI maps, the product follows the general rule of using CapitalCase after the product owner prefix. For example, **F1-AddToDoEntry** is the name of a base product business service.

NOTE: Data Explorer Business Services. For business services used to invoke a data explorer zone, it is recommended to name the Business Service the same name as the related zone rather than defining a different CapitalCase name for the business service.

Please note that this standard is followed for all new records introduced by the base product. However, there are base product entries in many of these system data tables that were introduced before the naming convention was adopted. That data does not follow the naming convention described above.

NOTE: Schema naming conventions. A context sensitive "Schema Tips" zone is associated with any page where a schema may be defined. The zone provides recommended naming conventions for elements within a schema along with a complete list of the XML nodes and attributes available to you when you construct a schema.

Implementation System Data

When new system data is introduced for your implementation you must consider the naming convention for the primary key. The product recommends prefixing records with **CM**, which is the value of the owner flag in your environment. This is consistent with the base product naming convention. This convention allows your implementation to use the CM packaging tool in the Software Development Kit as delivered. The extract file provided with the tool selects system data records with an owner flag of **CM AND** with a **CM** prefix.

NOTE: If you choose not to follow the CM naming convention for your records and you want to use the CM packaging tool, your implementation must customize the extract file to define the appropriate selection criteria for the records to be included in the package. Refer to the Software Development Kit documentation for more information.

Also note that owner flag may be introduced to an existing table in a new release. When this happens, the CM packaging tool is also updated to include these new system data tables. Your implementation will have existing records in those tables that probably do not follow any naming convention. After an upgrade to such a release, if you want to include this data in the CM packaging tool, you must customize the extract file for the tables in question.

Caching Overview

A great deal of information in the system changes infrequently. In order to avoid accessing the database every time this type of information is required by an end-user or a batch process, the system maintains a cache of static information on the web server and in the batch thread pool worker. These are referred to as the “application caches”. Some examples of application caches include

- System messages
- Field label and other field information
- Security Information

The framework product provides many specific caches for commonly used (and infrequently changed) data. In addition, specific edge applications may introduce additional caches as appropriate.

Information may also be cached on each user's browser.

The following topics highlight information about refreshing the various caches.

Server Cache

The server cache refers to data that is cached on the web server. An important use of this cache is for users' online access to the application. The caches aid in better performance while navigating throughout the system, allowing for data to be accessed from the cache rather than by always accessing the database. Besides user access to the web server cache, other functionality deployed to the web server uses caches in a similar way. For example, web services are deployed to the web server and access their own version of the cache.

The contents of the cache are cleared whenever the web server is restarted. This means that fresh values are retrieved from the database once users and web services start using the application again.

The product also supplies a flush command that one can issue in the browser's URL to immediately clear the contents of the cache. The command **flushAll.jsp** flushes every cache.

For example, assume the following:

- the web server and port on which you work is called **OU-Production:7500**
- you add a new record to a control table and you want it to be available on the appropriate transactions immediately

You would issue the following command in your browser's address bar: **http://OU-Production:7500/flushAll.jsp**. Notice that the command replaces the typical `cis.jsp` that appears after the port number.

If your system has been configured correctly, the **flushAll** command will submit a request to do a “global” flush of caches (including the web services cache and the thread pool worker cache). This functionality uses a JMS Topic to publish the flush request. Refer to the *Server Administration Guide* for details on how to configure the JMS topic.

Batch Cache

When submitting a batch job, the batch component uses a Hibernate data cache to cache administrative data that doesn't change very often. The tables whose records are included in this cache are configured using the Caching Regime value of **Cached for Batch**. Refer to [Table - Main](#) for more information. When starting a thread pool worker, data in tables marked as cached is loaded and cached for as long as that thread pool is running.

In addition batch jobs may also access application caches when applicable. When starting a thread pool worker, application data that is cached is loaded and cached for as long as that thread pool is running.

If there is a change in cached data that should be available for the next batch job, the following points highlight how the cache can be refreshed:

- By default the system is configured to automatically refresh the Hibernate cache every 60 seconds. However, an implementation may override the configuration to either change the number of seconds between intervals or to disable the automatic caching altogether. Application caches used by the batch jobs are not impacted by this refresh.
- Restart the thread pool workers.
- Run the **F1-FLUSH** (Flush all Caches) background process. This background process will flush the application data cached for all thread pool workers for all thread pools.
- If your the region has configured the thread pool workers to “listen” to requests for global flush as described in the [Server Cache](#) topic, the thread pool worker caches are also refreshed when a **flushAll** command is issued.

Client Cache

In addition to the web server's cache, information is also cached on each user's browser. After clearing the cache that's maintained on the web server, you must also clear the cache that's maintained on your client's browser. To do this, follow the following steps:

- Select **Tools** on your browser's menu bar
- Select **Internet Options...** on the menu that appears.
- Click the **Delete Files** button on the pop-up that appears.
- Turn on **Delete all offline content** on the subsequent pop-up that appears and then click **OK**.
- And then enter the standard URL to re-invoke the system.

NOTE: Automatic refresh of the browser's cache. Each user's cache is automatically refreshed based on the **maxAge** parameter defined in the web.xml document on your web server. We recommend that you set this parameter to **1** second on development / test environments and **28800** seconds (8 hours) on production environments. Please speak to system support if you need to change this value.

Expression Parser

The product provides support for defining expressions that may be of a mathematical or logical/boolean nature. The expression may include variables and functions.

The data explorer [column parameter](#) is an example of where this may be used. That parameter supports the definition of a formula. Edge applications may include support for a formula or expression using this parser as well. For example, several application include a type of ‘rule’ object (calculation rule, form rule or usage rule) that is used for validation or calculation that may support applying a formula.

The following tables highlight what is supported in the expressions that use this parser.

Category	Supported in Expression	Description
Data types	Number	
	String	
	Boolean	
	List	
Literals	Numbers	
	Strings surrounded with either single quote or double quote.	

Category	Supported in Expression	Description
	NOTE: 'Escaping' special characters is not currently supported.	
	Boolean values: true and false .	
Operations	+	Plus
	-	Minus
	/	Division
	*	Multiplication
	^ or **	Power
	%	Modulus
Logical operations	=	Equal
	>	Greater than
	>=	Greater than or equal to
	<	Less than
	<=	Less than or equal to
	!= or <>	Not equal to

This table identifies the functions that are supported. Note that several of the functions are applicable to a list of values. Note that although the functions are listed in lower case, the column parameter syntax in data explorer indicates referencing the functions as all capital letters. The system converts the data explorer column formula to lowercase before being evaluated.

Function	Description	Comments
size(list<element>)	Number of elements in the list.	
isEmpty(list<element>)	Returns true if the list is empty.	
sum(list<number element>)	Returns the sum of the numbers in the list.	
avg(list<number element>)	Returns the average of the numbers in the list.	
avg(num1, num2, ...)	Returns the average of the number arguments.	
max(list<comparable>)	Returns the largest value in the list.	
max(comparable1, comparable2, ...)	Returns the largest value of the number arguments.	
min(list<comparable>)	Returns the smallest value in the list.	
min(comparable1, comparable2, ...)	Returns the smallest value of the number arguments.	
abs(number)	Returns the absolute value.	
ceiling(number)	Rounds the number to the ceiling.	
exp10(number)	Raises 10 to the number power.	
acos(number)	Returns the arc cosine of the number in radians.	The result will lose precision based functions.
asin(number)	Returns the arc sine of the number radians.	The result will lose precision based functions.
atan(number)	Returns the arc tangent of the number radians.	The result will lose precision based functions.
cos(radians)	Returns the cosine of the radian angle input.	The result will lose precision based functions.
exp(number)	Raises e to the number power.	The result will lose precision based functions.
log10(number)	Takes the log, base 10, of the number.	The result will lose precision based functions.
log(number)	Takes the natural log (base e) of the number.	The result will lose precision based functions.
sin(radians)	Returns the sine of the radian angle input.	The result will lose precision based functions.

Function	Description	Comments
sqrt(number)	Returns the square root of the number.	The result will lose precision based functions.
tan(radians)	Returns the tangent of the radian angle input.	The result will lose precision based functions.
floor(number)	Rounds the number to the floor.	
round(number)	Assumes a scale of 0. The default rounding mode of "round half up" is applied.	
round(number, scale)	The default rounding mode of "round half up" is applied.	
round(number, scale, mode)	The mode must be set to one of the following: <ul style="list-style-type: none"> • "ROUND_CEILING" • "ROUND_DOWN" • "ROUND_FLOOR" • "ROUND_HALF_DOWN" • "ROUND_HALF_UP" • "ROUND_HALF_EVEN" • "ROUND_UP" • "ROUND_UNNECESSARY" 	
negate(number)	Returns the negative value of the number.	Only available in data entry

The following are special functions supported in the application for a list of values. In each case, the syntax is *function* [*indexVariable* in *listName* | *expression using indexVariable*], where the *indexVariable* is chosen by the formula writer to represent each entry in the list and the expression used to evaluate each entry must reference that variable.

NOTE: The syntax supported for a given use of the formula in a functional area is driven by that particular functional area. For example, in Oracle Public Sector Revenue Management, a formula is supported in the "conditional element validation" form rule. In that form rule all variables including lists are declared in the form rule using letters and the formulas in turn use these letters. In that scenario, the functions below would reference the declared variable letter as the "listName". Other specific functional area that use this expression parser may support different syntax for referencing elements or lists.

Function	Description	Examples
any [<i>indexVariable</i> in <i>listName</i> Boolean expression for <i>indexVariable</i>]	This function returns the value true if any of the entries in list satisfies the expression.	The following returns true if any value is greater than 0. <code>any [i in list]</code>
all [<i>indexVariable</i> in <i>listName</i> Boolean expression for <i>indexVariable</i>]	This function returns the value true if all of the entries in the list satisfy the expression.	The following returns true if all values are greater than 0. <code>all [i in list]</code>
collect [<i>indexVariable</i> in <i>listName</i> expression for <i>indexVariable</i>]	This function returns a new list of elements from the referenced list where the value of each entry of the new list is the result of the expression applied to each original value.	The following returns a list of the square amounts. <code>collect [i in list]</code>
select [<i>indexVariable</i> in <i>listName</i> Boolean expression for <i>indexVariable</i>]	This function returns a list of all the values of the original list that satisfy the Boolean expression.	The following returns a list of the positive numbers. <code>select [i in list]</code>
reject [<i>indexVariable</i> in <i>listName</i> Boolean expression for <i>indexVariable</i>]	This function returns a list of all the values of the original list that do not satisfy the Boolean expression.	The following returns a list of the negative numbers. <code>reject [i in list]</code>

Debug Mode

Your implementation team can execute the system using a special mode when they are configuring the application. To enable this mode, enter **?debug=true** at the end of the URL that you use to access the application. For example, if the

standard URL was `http://CD-Production:7500/cis.jsp`, you'd enter `http://CD-Production:7500/cis.jsp?debug=true` to enable configuration mode.

When in this mode certain debugging oriented tools become available right below the main toolbar.

- **Start Debug** starts a logging session. During this session the processing steps that you perform are logged. For example, the log will show the data areas that are passed in at each step and the data areas returned after the step is processed.
- **Stop Debug** stops the logging session.
- **Show Trace** opens a window that contains the logging session. All of the steps are initially collapsed.
- **Clear Trace** clears your log file.
- **Show User Log** allows you to view your own log entries. The number of "tail" entries to view may be specified in the adjacent **Log Entries** field before clicking the button. Limiting the number of entries to view allows the user to quickly and easily see only the latest log entries without having to manually scroll to the end of the log.
- Checking the **Global Debug** indication starts various tracing options.

Other parts of the system may show additional configuration oriented icons when in this mode. For example, explorer zones may provide additional tools to assist in debugging zone configuration. These icons are described in the context of where they appear.

Also, in debug mode drop down lists in data explorer and UI map zones will contain the code for each item in addition to the item's display string.

NOTE: Show User Log button is secured. An application service **F1USERLOG** has been provided for this functionality to allow implementations to restrict user access to this button. Such restriction may be called for in production environments.

System Override Date

The system provides a way to override the system date used for online operations. This feature is available if the server administrator has enabled it in the environment properties. For instructions on configuring environment properties see the *Server Administration Guide*. The system date override feature is not recommended for production environments.

Under the **General System Configuration** [Feature Configuration](#), the **System Override Date Option Type** holds the date the application will use as the global system date instead of retrieving the same from the database. This feature can be especially useful in running tests that require the system date to be progressed over a period of time.

The system override date feature is also available at the user level. This is useful when a user wants override the system date to run tests without affecting the system date for other users in the environment. In order to override the system date for the user, open the [User — Characteristics](#) page, add the **System Override Date** characteristic type with a characteristic value set to the desired date in the YYYY-MM-DD format.

If system override dates are defined at both the feature configuration level and the user level, the date set at the user level will take precedence.

Advanced Search Options

The product supports fuzzy searching in explorer zone types using the Oracle Text CONTAINS operator.

Refer to the DBA guide for details on setting up the database to support fuzzy searching. Note that there are some implementations where fuzzy searching will not be possible. For example, it's only available for implementations using the Oracle database. Additionally, not all languages are supported. Refer to the Oracle Database documentation for more information about fuzzy searching.

For information about the particular syntax to use in the explorer zones, refer to [SQL Statement](#) in the zone parameter details section.

To Do Lists

Certain events that occur within the system will trigger messages describing work that requires attention. For example, if a bill segment has an error, the system generates a To Do message to alert the person responsible for correcting such errors.

Each type of message represents a To Do list. For example, there are To Do lists for bill segment errors, payment errors, customer contact reminder, etc.

We refer to each message as a **To Do Entry**. Each To Do entry is assigned a specific **To Do Role**. The role defines the users who may work on the entry. A To Do entry has a **To Do log** that maintains record of the progress on the To Do entry. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed.

FASTPATH: Refer to [To Do Processing](#) for a description of end-user queries and tools assisting in reviewing, assigning and processing To Do entries.

The Big Picture of To Do Lists

The topics below provide more information about To Do configuration.

To Do Entries Reference A To Do Type

Every [To Do entry](#) references a To Do type. The To Do type controls the following functions:

- The To Do list on which the entry appears.
- The page into which a user is taken when they drill down on an entry.
- The message that appears in the user's To Do list. Note this message can be overridden for specific To Do messages by specifying a different message number in the process that creates the specific To Do entry. For example, the process that creates To Do entries associated with bill segments that are in error displays the error message rather than a generic "bill segment is in error" message.
- The To Do list's sort options. Sort options may be used on the To Do list page to sort the entries in a different order. For example, when a user looks at the bill segment error To Do list, they have the option of sorting it in error number order, account name order, or in customer class order. Note the default sort order is also defined on To Do type.
- Whether (and how) the To Do entry is downloaded to an external system (e.g., an email system).
- The roles to which an entry may be reassigned.
- The default priority of the To Do list in respect of other To Do lists.
- The To Do list's usage, which indicates whether a To Do of that type may be created manually by a user.
- The algorithms used to perform To Do list specific business rules.
- The characteristics applicable to the To Do list.

To Do Entries Reference A Role

Every [To Do entry](#) references a role. The role defines the users who may be assigned to **Open** entries.

The permissible roles that may be assigned to a To Do entry are defined on the entry's To Do type. After an entry is created, its role may be changed to any role defined as valid for the entry's To Do type.

An entry's initial role is assigned by the background process or algorithm that creates the entry. Because you can create your own processes and algorithms, there are an infinite number of ways to default an entry's role. However, the base package processes and algorithms use the following mechanisms to default an entry's role:

- The system checks if an entry's message category / number is suppressed (i.e., not created). If so, the entry is not created. Refer to [To Do Entries Can Be Rerouted Or Suppressed Based On Message Number](#) for more information.
- The system checks if an entry's message category / number is rerouted to a specific role. If so, it defaults this role. Refer to [To Do Entries Can Be Rerouted Or Suppressed Based On Message Number](#) for more information.
- Your specific product may introduce additional criteria for assigning a role, for example perhaps important accounts are assigned to a kind of account management group and the account management group includes configuration for special role for certain To Do types. Refer to [Set Additional Information Before a To Do is Created](#) for more information.
- If a Role wasn't determined in one of the previous steps and a Role is provided by the initiating process, the entry is created with that Role.
- If the entry does not have a role after the above takes place, the entry's To Do type's default role is assigned to the entry.

NOTE:

At installation time, the system provides a default role assigned to the system To Do types when first installed called **F1_DFLT**. This is done to allow testing of the system prior to implementing of appropriate To Do roles for your organization. The recommendation is to configure all the To Do Types with appropriate business oriented To Do roles once they are defined.

CAUTION: Important! Most organizations have the notion of a supervisor who is responsible for all entries assigned to a given role. It's important for this user (or users) to be part of all such roles. Refer to [To Do Supervisor Functions](#) for information about transactions that can be used by supervisors to review and assign work to users.

To Do Entries Can Be Rerouted (Or Suppressed) Based On Message Number

Consider the To Do type used to highlight bill segments that are in error. To Do entries of this type reference the specific bill segment error number so that the error message can be shown when the Bill Segments in Error To Do list is displayed.

NOTE: Message Category / Message Number. Every error in the system has a unique message category / number combination. Refer to [The Big Picture of System Messages](#) for more information about message categories and numbers.

If you want specific types of errors to be routed to specific users, you can indicate such on the To Do type. For example, if certain bill segment errors are always resolved by a given rate specialist, you can indicate such on the To Do type. You do this by updating the [To Do type's message overrides](#). On this page you specify the message category / number of the error and indicate the To Do role of the user(s) who should work on such errors. Once the To Do type is updated, all new To Do entries of this type that reference the message number are routed to the desired role.

NOTE: Reroute versus suppression. Rather than reroute an entry to a specific role, you can indicate that an entry with a given message number should be suppressed (i.e., not created). You might want to do this if you have a large volume of certain types of errors and you don't want these to clutter your users' To Do lists.

Obviously, you would only reroute those To Do types that handle many different types of messages. In other words, if the To Do type already references a specific message category / number rerouting is not applicable.

We do not supply documentation of every possible message that can be handled by a given To Do type. The best way to build each To Do type's reroute list is during the pre-production period when you're testing the system. During this period, compile a list of the messages that should be routed to specific roles and add them to the To Do type.

Keep in mind that if a message number / category is not referenced on a To Do type's reroute information, the entry is routed as described under [To Do Entries Reference A Role](#).

NOTE: Manually created To Do entries cannot be rerouted or suppressed. The rerouting occurs as part of the batch process or algorithm processing when the To Do is created. The role or user to whom a manual To Do should be assigned is specified when the To Do is created online. A manually created To Do may also be forwarded to another user or role.

The Priority Of A To Do Entry

Some To Do entries may be more urgent to resolve than others. A To Do entry is associated with a priority level representing its relative processing order compared to other entries.

Priority level is initially assigned as follows:

- If a **Calculate Priority** plug-in is defined on the To Do entry's type, the system calls it to determine the entry's priority. You may want to use this method if an entry's priority is based on context or time-based factors. For example, when priority takes into consideration account specific attributes. When applicable, you may design a process that triggers priority recalculation of To Do entries "at will". For example, when priority is reassessed periodically based on factors external to the To Do entry's information. Refer to [To Do Type](#) for more information on priority calculation algorithms.
- If a priority value has not been determined by a plug-in, the system defaults a To Do entry's initial priority to the value specified on its type.

A user may manually override a To Do entry's priority at any time. Notice that once a To Do entry's priority is overridden, **Calculate Priority** plug-ins are no longer called so as to not override the value explicitly set by the user.

NOTE: The system does not use priority values to control order of assignment nor processing of To Do entries. Priority is available to assist your organization with supporting a business practice that ensures higher priority issues are worked on first.

Working On A To Do Entry

A user can drill down on a To Do entry. When a user drills down on an entry, the user is transferred to the transaction associated with the entry. For example, if a user drills down on a bill segment error entry, the user is taken to the Bill Segment - Main page. Obviously, the page to which the user is taken differs depending on the type of entry.

It is also possible to configure the To Do type to launch a [script](#) when a user drills down on an entry rather than taking the user to a transaction. The script would walk the user through the steps required to resolve the To Do entry. Refer to [Launching Scripts When A To Do Is Selected](#) for more information.

After finishing work on an entry, the user can mark it as **Complete**. Completed entries do not appear on the To Do list queries (but they are retained on the database for audit purposes). If the user cannot resolve the problem, the user can forward the To Do to another user.

Launching Scripts When A To Do Is Selected

Users can complete many To Do entries without assistance. However, you can set up the system to launch a [script](#) when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

A script is linked to a To Do type based on its message number using the [To Do type's message overrides](#). Refer to [Executing A Script When A To Do Is Selected](#) for more information.

To Do Entries Have Logs

Each [To Do entry](#) has a To Do log that maintains a record of the To Do's progress in the system. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed. Users can view the log to see who assigned them a particular To Do and whether any work has already been done on the To Do.

A log entry is created for all actions that can be taken on a To Do entry. Log entries are created for the following events:

- A To Do entry is created (either by the system or by a user)
- A To Do entry is completed (either by the system or by a user)
- A user takes an open To Do entry
- A supervisor assigns a To Do entry
- A user forwards an entry to another user or role
- A user sends back a To Do to the user who forwarded it
- A user manually adds a log entry to record details about the To Do's progress
- A user manually overrides the To Do entry's priority

FASTPATH: For information about the contents of log entries for each of the events, refer to [Log Entry Events](#).

How Are To Do Entries Created?

A To Do Entry may be created in the following ways:

- A [background process](#) can create To Do Entries.
- An [algorithm](#) can create entries of a given type. Because the use of algorithms is entirely dependent on how you configure the control tables, the number of types of such entries is indeterminate.
- A user can create entries of To Do types that have a **Manual** usage. Refer to [To Do Entries Created Manually](#) for information about setting up manual To Do types.

For any base product process that includes logic to create a To Do entry, the system supplies a sample To Do type that may be used. Although the To Do types provided by the product are system data, the following information related to each To Do type may be customized for an implementation and is not overwritten during an upgrade:

- The creation process. If the To Do is created by a background process where the background process is referenced on a To Do type. Refer to [To Do Entries Created By Background Processes](#) for more information.
- The routing process. Refer to [To Do Entries May Be Routed Out of the System](#) for more information.
- The priority. Refer to [To Do Type - Main](#) for more information.
- The [roles](#) that may be associated with the To Do type. Refer to [To Do Entries Reference a Role](#) for more information.
- The [message override](#) information. Refer to [To Do Entries Can Be Rerouted \(Or Suppressed\)](#) and [Launching Scripts When a To Do Is Selected](#) for more information.

To Do Entries Created By Background Processes

There are different types of To Do entries created by background processes:

- To Do entries created by dedicated To Do background processes
- To Do entries created for object-specific errors detected in certain background processes

- To Do entries created based on a specific condition

Dedicated To Do Background Processes

There are To Do entries that are created by system background processes whose main purpose is to create To Do entries based on a given condition. For these background processes, the To Do Type indicates the creation background process.

NOTE: If you don't schedule the background process, the entries will NOT be created! The To Do entries of this type will only be created if you have scheduled the associated background process. Therefore, if you want the system to produce a given entry, schedule the background process.

To Dos Created for Object-Specific Error Conditions

A system background process may create a To Do entry when an error is detected during object-specific processing. This is applicable for processes that do not have built in error handling, for example where there is an explicit "error" state or where the record has an explicit "exception" record.

For these background processes, the To Do Type must reference the creation background process.

To have the system create To Do entries for some or all of the errors generated by one of these processes, you must do the following:

- If you want the system to generate To Do entries for errors detected by one of the background processes below, go to the appropriate To Do type and populate the creation background process.
- If you want the system to generate To Do entries for some errors for the process, but not for all errors, populate the creation background process and then proceed to the [message overrides](#) tab to [suppress](#) certain messages. To this by indicating the message category and message number you want to suppress. Any error that is suppressed is written to the [batch run tree](#).

The functionality will only create a new To Do entry if there is not already an existing (non-complete) To Do for the same To Do type and drill key. It will also check for an existing To Do for a successfully processed record and complete that To Do.

If you do not populate the creation background process, the errors are written to the [batch run tree](#).

NOTE: Errors received while creating a To Do entry. If the background process cannot successfully create a To Do entry to report an object-specific error, the error is written to the batch run tree along with the error encountered while attempting to create the To Do entry.

NOTE: System errors are not included. To Do entries are not created for a system error, for example an error related to validation of input parameter. These errors are written to the [batch run tree](#). Refer to [Processing Errors](#) for more information.

To Dos Created by Background Processes for Specific Conditions

There are some system background processes that create a To Do entry when the process detects a specific condition that a user should investigate. For each background process, the To Do type is an input parameter to the process. The system provides To Do types for each base package background process that may create a To Do entry.

NOTE: No Creation Process. These To Do types do not need (and should not have) a **Creation Process** specified.

To Do Entries Created By Algorithms

There are To Do entries that are created by algorithm types supplied with the base package. The system supplies a To Do Type for each of these To Do entries that you may use.

If you want to take advantage of these types of entries for system algorithm types, you must do the following:

- Create an [algorithm](#):
 - This algorithm must reference the appropriate Algorithm Type.
 - These algorithms have a parameter of the To Do Type to be created. You should specify the To Do Type indicated in the table.
- Plug the algorithm into the respective control table.

To Do Entries Created Manually

You must set up manual To Do entry types if you want your users to be able to create To Do entries online. Users may create a manual To Do entry as a reminder to themselves to complete a task. Online To Do entries may also be used like electronic help tickets in the system. For example, if a user is having a problem starting service, the user can create a To Do that describes the problem. The To Do can be assigned to a help resolution group that could either resolve the problem or send the To Do back to the initiating user with information describing how to resolve the problem.

If you want to take advantage of manual To Do entries, create a To Do type and specify the following information.

On the Main tab:

- Set the **To Do Type Usage** flag to **Manual**.
- Set the **Navigation Option** to **toDoEntryMaint** (To Do entry maintenance).
- Set the **Message Category** and **Message Number** to the message you want to be used for To Do entries of this type. The system will populate the message parameter with the Subject. To show only the subject in the To Do's message, use a message with "%1" as its text.

On the Roles tab:

- Specify the [To Do roles](#) that may be assigned to To Do entries of this type.
- Indicate the To Do role that should be defaulted when you create To Do entries of this type.

On the Sort Keys tab:

When a user adds a manual To Do entry, the system creates an entry with three sort key values. (Sort keys may be used on the To Do list page to sort the entries in a different order.) The To Do type should be set up to reference the sort keys as follows:

Sequence	Description
1	Created by user ID
2	Created by user name
3	Subject

We recommend that the keys have an **Ascending** sort order and that the Subject is made the default sort key.

NOTE: It is possible to define additional sort keys and use a To Do Post Processing algorithm to populate the values. In this case, the base sort keys defined above should still be defined.

On the Drill Keys tab:

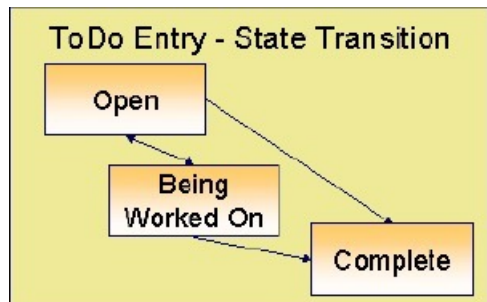
When a user adds a manual To Do entry, it is created with a drill key value equal to the To Do entry's ID. When the user clicks the Go To button next to the message in the To Do list, the system uses the drill down application service (defined on the main tab) and the drill key to display the associated To Do entry.

The To Do type must be set up with a drill key that reference the To Do entry table and the To Do entry ID:

Sequence	Table	Field
1	CI_TD_ENTRY	TD_ENTRY_ID

The Lifecycle Of A To Do Entry

The following state transition diagram will be useful in understanding the lifecycle of a To Do entry.



A To Do entry is typically created in the **Open** state. Entries of this type can be viewed by all users belonging to the entry's role. Refer to [How Are To Do Entries Created?](#) for information about how entries are created.

An **Open** entry becomes **Being Worked On** when it is assigned to a specific user or when a user proactively assumes responsibility for the entry. While an entry is **Being Worked On**, it is visible on the To Do Summary page only by the user who is assigned to it.

NOTE: To Do entries may be created in the **Being Worked On** state. Some To Do background processes may create To Do entries in the **Being Worked On** state. When a user adds a To Do entry online and assigns the To Do to a user (as opposed to a role), the To Do entry is also created in the **Being Worked On** state.

A **Being Worked On** entry may be forwarded to a different user or role. If the entry is forwarded to a role, it becomes **Open** again.

When an entry becomes **Complete**, it is no longer visible in the To Do list queries (but it remains on the database for audit purposes). There are two ways an entry can become **Complete**:

- A user can manually indicate it is **Complete** (there are several ways to do this).
- For To Do entries that are logically associated with the state of some object, the system automatically marks the entry **Complete** when the object is no longer in the respective state. For example, an entry that's created when an account doesn't have a bill cycle is completed when the account has a bill cycle.

CAUTION: Important! The automatic completion of To Do entries occurs when the background process responsible for creating entries of a given type is executed. Therefore, if you only run these processes once per day, these entries remain **Being Worked On** even if the object is no longer in the respective state.

Linking Additional Information To A To Do Entry

Additional information may be linked to a To Do entry using characteristics. For example, when creating a manual To Do entry, a user may define the account related to the To Do.

When creating an automatic To Do entry, the program that generates the To Do may link related data to the To Do using characteristics. Use system [algorithm](#) to link related entities. For manually created To Dos, the valid characteristic types that may be linked to the To Do entry must be defined on the [To Do type](#) for that To Do entry.

If your To Do entries reference characteristics that are related to your global context data, you may want to configure an [Alerts](#) to display an alert if a related entry is **Open** or **Being Worked On**.

Implementing Additional To Do Entry Business Rules

If your business practice calls for additional validation rules or processing steps to take place after a To Do Entry is created or updated, you may want to take advantage of the **To Do Post Processing** plug-ins defined on [To Do type](#).

For example, you may want to validate that To Do entries are only assigned to users with the proper skill levels needed to resolve them. Refer to [F1-VAL-SKILL](#) for a sample algorithm handling such validation.

To Do Entries May Be Routed Out Of The System

A To Do type can be configured so that its entries are interfaced to another system.

For example, a given To Do type can be configured to create an email message whenever a new To Do entry is created. The following points describe how to do this:

- Define the name of the background process responsible for interfacing the new To Do entries to another system on the respective To Do type. The base package contains a batch process called [F1-TDEER](#) that can be used for most situations. This batch process invokes the **External Routing algorithms** defined on each entry's To Do type.
- Plug in an appropriate **External Routing** algorithm on the respective To Do type. The logic in this type of algorithm performs the interface efforts for a specific To Do entry. For example, if an email message should be created for a To Do entry, the logic in the algorithm would compose and send the email message(s) for a specific To Do entry.

Click [here](#) to see the algorithm types available for this system event.

Periodically Purging To Do Entries

Completed To Do entries should be periodically purged from the system by executing the [F1-TDPG](#) background process. This background process offers you the following choices:

- You can purge all To Do entries older than a given number of days.
- You can purge To Do entries for a specific list of To Do types that are older than a given number of days.
- You can purge all To Do entries except for a specific list of To Do types that are older than a given number of days.

We want to stress that there is no system constraint as to the number of **Completed** To Do entries that may exist. You can retain these entries for as long as you desire. However, you will eventually end up with a very large number of **Completed** entries and these entries will cause the various To Do background processes to degrade over time. Therefore, you should periodically purge **Completed** To Do entries as they exist only to satisfy auditing and reporting needs.

NOTE: Different retention periods for different types of To Do entries. Keep in mind that the purge program allows you to retain different types of entries for different periods of time.

Setting Up To Do Options

The topics in this section describe how to set up To Do management options.

Installation Options

The following section describes configuration setup on the installation options.

To Do Information May Be Formatted By An Algorithm

A **To Do Information** algorithm may be plugged in on the [installation record](#) to format the standard To Do information that appears throughout the system. This algorithm may be further overridden by a corresponding plug-in on the [To Do Type](#).

Set Additional Information Before A To Do Is Created

A **To Do Pre-creation** algorithm may be plugged in on the [installation record](#) to set additional information for a To Do entry before it is created. Algorithms of this type are used for two common purposes:

- Linking context specific data to the To Do entry using characteristics. For example, Oracle Utilities Customer Care and Billing provides an algorithm that attempts to link a related person, account, premise, service agreement or service point to a To Do entry based on its drill key value. Note, before you can set up this algorithm, you must define the characteristic types that you'll use to hold each of these entities. Also note that it is not necessary to define these characteristics as valid characteristic types on the To Do type.
- Overriding the Role of a To Do entry based on specific configuration related to the To Do's context data. For example, Oracle Utilities Customer Care and Billing provides an algorithm to determine a To Do role based on overrides related to the account's account management group or division.

Alerts

If your To Do entries reference characteristics related to your global context data and your product supports dashboard alerts generated by algorithms, you may want configure an algorithm to display an alert if the entry is **Open** or **Being Worked On** for the data currently in context.

Refer to your product's documentation to determine if these types of alerts are supported.

Next Assignment Algorithm

If your organization opts to use the next assignment feature supported by the Current To Do dashboard zone, you need to plug-in a **Next To Do Assignment** algorithm into the [installation options](#) to determine the next To Do entry the user should work on. Make sure you provide users with security access rights to the zone's next assignment action.

FASTPATH: Refer to the [Current To Do](#) zone for more information.

Messages

You need only set up new messages if you use algorithms to create To Do entries or prefer different messages than those associated with the base package's To Do types.

Feature Configuration

The base package is provided with a generic **Activity Queue Management Feature Configuration** type. You may want to set up a feature configuration of this type to define any To Do management related options supporting business rules specific to your organization.

For example, the base package provides the following plug-ins to demonstrate a business practice where To Do entries are only assigned to users with the proper skill levels to work on them.

- The base **To Do Post Processing** To Do Type algorithm **F1-VAL-SKILL** validates that a user has the necessary skill levels required to work on a given To Do entry.
- The base **Next To Do Assignment** installation options algorithm **F1-NEXT-ASSG** only assigns To Do entries to users that have the proper skills to work on them. This plug-in is only applicable if your organization practices **work distribution** "on demand."

You must set up such an **Activity Queue Management** feature configuration if you want to use any of the above base package plug-ins.

The following points describe the various **Option Types** provided with the base package:

- **Skill.** This option provides a reference to a skill category. For example, if you were using characteristics to represent skill categories then you should reference each characteristic type using this option.
- **Override Skill.** This option provides an override skill information reference for a specific message. For example, if you were using a To Do Type characteristic to specify an override skill category and level for a specific message category / number then you would reference this characteristic type using this option.

NOTE: Skill Setup. Refer to the description of the above base package algorithms for further details on how to setup skill level information.

NOTE: More Options. Your implementation may define additional options types. You do this by add new lookup values to the lookup field **F1QM_OPT_TYP_FLG**.

NOTE: Only one. The system expects only one **Activity Queue Management** feature configuration to be defined.

Defining To Do Roles

This section describes the control table used to maintain To Do roles.

To Do Role - Main

The **Main** page is used to define basic information about a To Do role.

To maintain this information, select **Admin > General > To Do Role**.

Description of Page

Enter a unique **To Do Role** and **Description** for the To Do role.

The grid contains the ID of each **User** that may view and work on entries assigned to this role. The First Name and Last Name associated with the user is displayed adjacent.

NOTE: System Default Role. The system supplies a default role **F1_DFLT** linked to each system To Do type. This is done so that To Do functionality may be tested prior to the creation of appropriate business oriented To Do roles.

Where Used

Follow this link to view the tables that reference [CI_ROLE](#) in the data dictionary schema viewer.

In addition, various “type” objects or algorithms may reference a To Do role to use when creating a To Do for a given business scenario. This is dependent on your specific product.

To Do Role - To Do Types

The **To Do Types** page defines the To Do types that may be viewed and worked on by users belonging to a given To Do role.

To maintain this information, select **Admin > To Do Role > Search** and navigate to the **To Do Types** page.

Description of Page

Enter the ID of each **To Do Type** whose entries may be viewed and worked on by the role.

Use As Default is a display-only field that indicates if the role is assigned to newly created entries of this type. You may define the default role for a given To Do type on the To Do Type maintenance page.

CAUTION: If you remove a To Do type where this role is the default, you must define a new role as the default for the To Do type. You do this on the To Do Type maintenance page.

Defining To Do Types

This section describes the control table used to maintain To Do types.

To Do Type - Main

The **Main** page is used to define basic information about a To Do type.

FASTPATH: Refer to [The Big Picture Of To Do Lists](#) for more information about To Do types and To Do lists in general.

To maintain this information, select **Admin > General > To Do Type**.

CAUTION: Important! If you introduce a To Do type, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Description of Page

Enter a unique **To Do Type** and **Description** for the To Do type.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

Use the **Detailed Description** to provide further details related to the To Do Type.

Enter the default **Priority** of To Do entries of this type in respect of other To Do types. Refer to [The Priority Of A To Do Entry](#) for more information.

For **To Do Type Usage**, select **Automatic** if To Dos of this type are created by the system (i.e., a background process or algorithm). Select **Manual** if a user can create a To Do of this type online.

Define the **Navigation Option** for the page into which the user is transferred when drilling down on a To Do entry of this type.

Use **Creation Process** to define the background process, if any, that is used to manage (i.e., create and perhaps complete) entries of this type. A **Creation Process** need only be specified for those To Do types whose entries are created by a background process. Refer to [To Do Entries Created By Background Processes](#) for more information.

Use **Routing Process** to define the background process that is used to download entries of a given type to an external system, if any. A **Routing Process** need only be specified for those To Do types whose entries are routed to an external system (e.g., an Email system or an auto-dialer). Refer to [To Do Entries May Be Routed Out Of The System](#) for more information.

Use **Message Category** and **Message Number** to define the message associated with this To Do type's entries. Note: this message will only be used if the process that creates the To Do entry does not supply a specific message number. For example, the process that creates To Do entries that highlight bill segments that are in error would not use this message; rather, the entries are marked with the message associated with the bill segment's error.

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all To Do entries of this type. You may enter more than one characteristic row for the same characteristic type, each associated with a unique **Sequence** number. If not specified, the system defaults it to the next sequence number for the characteristic type.

Where Used

Follow this link to view the tables that reference [CI_TD_TYPE](#) in the data dictionary schema viewer.

To Do Type - Roles

The **Roles** page defines the roles who may view and work on entries of a given To Do type.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Roles** page.

Description of Page

Enter each **To Do Role** that may view and work on entries of a given type. Turn on **Use as Default** if the role should be assigned to newly created entries of this type. Only one role may be defined as the default per To Do type.

FASTPATH: Refer to [To Do Entries Reference A Role](#) for more information about roles and To Do entries.

To Do Type - Sort Keys

The **Sort Keys** page defines the various ways a To Do list's entries may be sorted. For example, when you look at the bill segment error To Do List, you have the option of sorting the entries in error number order, account name order, or in customer class order.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Sort Keys** page.

CAUTION: Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

Description of Page

The following fields display for each sort key.

Sequence is the unique ID of the sort key.

Description is the description of the sort key that appears on the To Do list.

Use as Default indicates the default sort key (the one that is initially used when a user opens a To Do list). Only one sort key may be defined as the default per To Do type.

Sort Order indicates whether the To Do entries should be sorted in **Ascending** or **Descending** order when this sort key is used.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

To Do Type - Drill Keys

The **Drill Keys** page defines the keys passed to the application service (defined on the Main page) when you drill down on an entry of a given type.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Drill Keys** page.

CAUTION: Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

Description of Page

Navigation Option shows the page into which the user is transferred when drilling down on a To Do entry of this type.

The following fields display for each drill key.

Sequence is the unique ID of the drill key.

Table and **Field** are passed to the application service when you drill down on an entry of a given type.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

To Do Type - Message Overrides

The **Message Overrides** page is used if you want To Do entries that reference a given message category / number to be routed to a specific To Do role (or suppressed altogether) or if you want to associate a script to a given message category / number.

FASTPATH: Refer to [To Do Entries Reference A Role](#) and [To Do Entries Can Be Rerouted Or Suppressed](#) for more information.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Message Overrides** page.

Description of Page

The following fields display for each override.

Message Category and **Number** allow the message to be overridden.

Exclude To Do Entry indicates if a To Do entry of this type that references the adjacent **Message Category** and **Number** should NOT be created.

Override Role indicates the to do role to which a To Do entry of this type that references the adjacent **Message Category** and **Number** should be addressed. This field is protected if **Exclude To Do Entry** is on.

Script indicates the script that should execute when a user drills down on a To Do entry of this type that references the adjacent **Message Category** and **Number**. This field is protected if **Exclude To Do Entry** is on. Refer to [Working On A To Do Entry](#) for more information.

To Do Type - To Do Characteristics

The **To Do Characteristics** page defines characteristics that can be defined for To Do entries of this type. The characteristic types for characteristics that are linked to the To Do entry as a result of a pre-creation algorithm do not need to be defined here.

To maintain this information, select **Admin > General > To Do Type > Search** and navigate to the **To Do Characteristics** page.

Turn on the **Required** switch if the **Characteristic Type** must be defined on To Do entries of this type.

Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

To Do Type - Algorithms

The **To Do Algorithms** page defines the algorithms that should be executed for a given To Do type.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Algorithms** page.

Description of Page

The grid contains **Algorithms** that control important To Do functions. If you haven't already done so, you must [set up the appropriate algorithms](#) in your system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Calculate Priority	Optional	<p>Algorithms of this type may be used to calculate a To Do entry's priority. They are called initially when a To Do entry is created and each time it gets updated so long as the To Do entry's priority has not been manually overridden. Once overridden, these algorithms are not called anymore.</p> <p>Note that it is not the responsibility of the algorithms to actually update the To Do entry with the calculated priority value but rather only return the calculated value. The system carries out the update as necessary.</p> <p>If more than one algorithm is plugged-in the system calls them one by one until the first to return a calculated priority.</p> <p>Click here to see the algorithm types available for this system event.</p>
External Routing	Optional	<p>Algorithms of this type may be used to route a To Do entry to an external system.</p> <p>The base package F1-TDEER background process invokes the algorithms for every To Do entry that its type references the process as the Routing Process and that the entry was not already routed. The background process marks an entry as routed by updating it with the batch control's current run number.</p> <p>If more than one algorithm is plugged-in the batch process calls them one by one until the first to indicate the To Do entry was routed.</p> <p>Click here to see the algorithm types available for this system event.</p>
To Do Information	Optional	<p>We use the term "To Do information" to describe the basic information that appears throughout the system to describe a To Do entry. The data that appears in "To Do information" is constructed using this algorithm.</p> <p>Plug an algorithm into this spot to override the "To Do information" algorithm on installation options or the system default "To Do</p>

System Event	Optional / Required	Description
		information" if no such algorithm is defined on installation options. Click here to see the algorithm types available for this system event.
To Do Post-Processing	Optional	Algorithms of this type may be used to validate and/or further process a To Do entry that has been added or updated. Click here to see the algorithm types available for this system event.

List of System To Do Types

The To Do types available to use with the product are found in the To Do Type page. In addition, they may be viewed in the [application viewer's To Do type](#) viewer. If your implementation adds To Do types, you may [regenerate](#) the application viewer to see your additions reflected there.

Implementing The To Do Entries

To enable the To Do entries visible in the To Do Type page and application viewer, you must configure the system as follows:

- Define the To Do roles associated with each To Do type and link the appropriate users to them. Once you have defined the roles appropriate for your organization's To Do types, remove the reference to this system default role **F1_DFLT**. Refer to [To Do Entries Reference A Role](#) for more information.
- For any To Do Type that is provided for a specific background process, the To Do simply needs to reference the appropriate Creation Background Process. When the background process is scheduled, To Dos are created based on the logic of the related background process. This applies to [To Dos Created for Object-Specific Error Conditions](#) and [Dedicated To Do Background Processes](#).
- For any To Do Type that is provided for creation by an algorithm or other process, there may be configuration required to populate that To Do type as an algorithm parameter or as an attribute on a control table.

NOTE: Refer to the description of the To Do type for more information.

Background Processes

This chapter covers various topics related to background processes. Besides providing an overview of background process functionality, the various tools available within the application to define, submit and monitor background processes are covered.

NOTE: Your specific source application may have additional background process topics. Please refer to the documentation section that applies to your source application for more information.

Understanding Background Processes

This section describes various topics related to the background processes that perform many important functions throughout your product such as:

- Processing To Do Entries
- Monitor processes that select records in a given state to progress them to their next state in their lifecycle

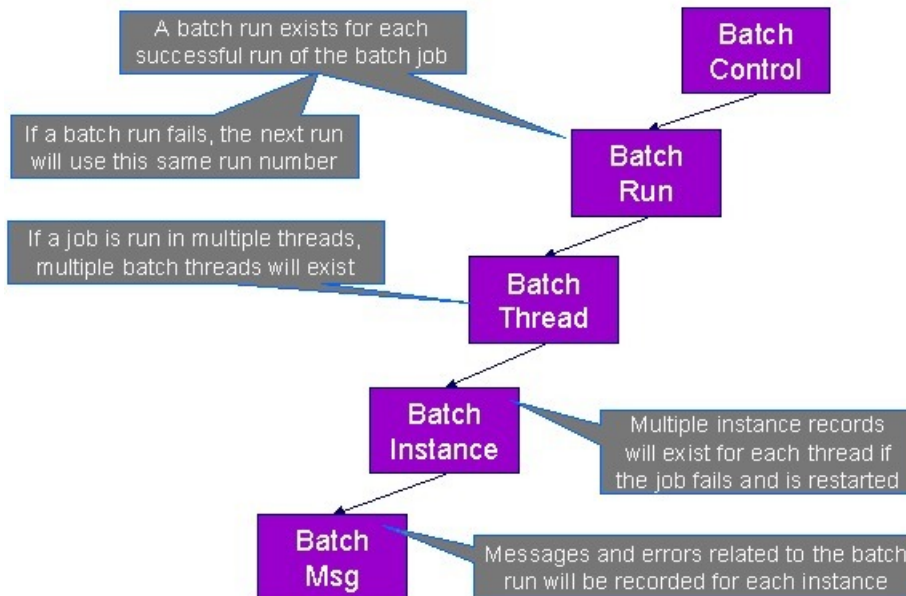
- Processes that purge data
- Processes that extract data
- And many more...

Background Processing Overview

While the system relies on a scheduler to secure and execute its background processes, there are additional issues that you should be familiar with:

- Batch control records are used for the following purposes:
 - Define the code that executes the logic associated with the background process.
 - For processes that extract information, the batch control record defines the next batch number to be assigned to new records that are eligible for extraction. For example, the batch control record associated with the process that routes To Do entries to an external system defines the next batch number to be assigned to new To Do entries that are configured with this batch control. When this To Do external routing process next runs, it selects all To Do entries marked with the current batch number (and increments the next batch number).
 - The batch control record for each background process organizes audit information about the historical execution of the background process. The system uses this information to control the restart of failed processes. You can use this information to view error messages associated with failed runs.
 - Many processes have been designed to run in parallel in order to speed execution. For example, the process that applies updates for a migration data set import for CMA can be executed so that multiple "threads" are processing a different subset of records (and multiple threads can execute at the same time). Batch control records associated with this type of process organize audit information about each thread in every execution. The system uses this information to control the restart of failed threads. Refer to [Parallel Background Processes](#) for more information.
 - Some processes define extra parameters. These parameters are defined with the batch control. Default values may also be captured for each parameter. They will be used when the [background process is submitted on-line](#).

The following diagram illustrates the relationships that exist for batch control records.



Results of each batch run can be viewed using the [Batch Run Tree](#) page.

Refer to [Batch Scheduler Integration](#) for information about the integration with the Oracle Scheduler.

Parallel Background Processes

Many processes have been designed to run in parallel in order to speed execution. This is referred to as running the process with multiple “threads”.

The system provides two strategies for distributing the data to the multiple threads.

- **Thread Level SQL Select.** This strategy is sometimes referred to as the “thread iterator” strategy. In this strategy, the batch job uses the primary key to figure out how to evenly distribute key ranges to each thread. Each thread is then responsible for selecting the records. In this strategy, the threads should also re-select the data periodically to release the cursor, which aids in performance. Note that this strategy is preferred but may only be used under the following conditions:
 - The data from only one maintenance object is being processed.
 - The primary key for the maintenance object is a single, numeric system generated key.

NOTE: Parameters may be used to override the low and high id. Refer to [Parameters Supplied to Background Processes](#) for more information.

- **Job Level SQL Select.** This strategy is sometimes referred to as the “standard commit” strategy. In this strategy, the keys for the records to be processed by the batch job are all selected first and stored in a temporary table. The batch job then supplies each thread with a range of keys that it should process. This strategy is used if multiple maintenance objects are being processed by the batch job; if the primary key of the maintenance object has multiple parts or if the primary key is non-numeric.

The multi-threading logic relies on the fact that primary keys for master and transaction data are typically system generated random keys. In addition, if the data is partitioned, it is expected to be partitioned based on the primary key.

NOTE: The detailed description in the metadata for each batch control provided with the system should indicate if it may be run in parallel. Note that the strategy used is not typically indicated in the detailed description.

NOTE: Overriding the thread ranges. Your implementation has the ability to override the thread ranges if certain data in your system takes longer to process. For example, imagine you have a single account in Oracle Utilities Customer Care and Billing that has thousands of service agreements (maybe the account for a large corporation or a major city). You may want to set up the thread ranges to put this large account into its own thread and distribute the other accounts to the other threads. To do this, you should create the appropriate batch thread records ahead of time in a status of **Thread Ready (50)** with the key ranges pre-populated. Note that the base product does not provide the ability to add batch thread records online. If you are interested in more information about this technique, contact Customer Support.

Optimal Thread Count

Running a background process in multiple threads is almost always faster than running it in a single thread. The trick is determining the number of threads that is optimal for each process.

NOTE: A good rule of thumb is to have one thread for every 100 MHz of application server CPU available. For example if you have four 450 MHz processors available on your application server, you can start with 18 threads to begin your testing: $(450 * 4) / 100 = 18$.

This is a rule of thumb because each process is different and is dependent on the data in your database. Also, your hardware configuration (i.e., number of processors, speed of your disk drives, speed of the network between the database server and the application server) has an impact on the optimal number of threads. Please follow these guidelines to determine the optimal number of threads for each background process:

- Execute the background process using the number of threads dictated by the rule of thumb (described above). During this execution, monitor the utilization percentage of your application server, database server and network traffic.
- If you find that your database server has hit 100% utilization, but your application server hasn't one of the following is probably occurring:
 - There may be a problematic SQL statement executing during the process. You must capture a database trace to identify the problem SQL.
 - It is also possible that your commit frequency may be too large. Commit frequency is a parameter supplied to every background process. If it is too large, the database's hold queues can start swapping. Refer to [Parameters Supplied to Background Processes](#) for more information about this parameter.
- It is normal if you find that your application server has hit 100% utilization but your database server has not. This is normal because, in general, all processes are CPU bound and not IO bound. At this point, you should decrease the number of threads until just under 100% of the application server utilization is achieved. And this will be the optimal number of threads required for this background process.
- If you find that your application server has NOT hit 100% utilization, you should increase the number of threads until you achieve just under 100% utilization on the application server. And remember, the application server should achieve 100% utilization before the database server reaches 100% utilization. If this proves not to be true, something is probably wrong with an SQL statement and you must capture an SQL trace to determine the culprit.

Another way to achieve similar results is to start out with a small number of threads and increase the number of threads until you have maximized throughput. The definition of "throughput" may differ for each process but can be generalized as a simple count of the records processed in the batch run tree. For example, in the Billing background process in Oracle Utilities Customer Care and Billing, throughput is the number of bills processed per minute. If you opt to use this method, we recommend you graph a curve of throughput vs. number of threads. The graph should display a curve that is steep at first but then flattens as more threads are added. Eventually adding more threads will cause the throughput to decline. Through this type of analysis you can determine the optimum number of threads to execute for any given process.

Parameters Supplied To Background Processes

This section describes the various types of parameters that are supplied to background processes.

General Parameters

The following information is passed to every background process.

- **Batch code.** Batch code is the unique identifier of the background process.
- **Batch thread number.** Thread number is only used for background processes that can be run in multiple parallel threads. It contains the relative thread number of the process. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20). Refer to [Optimal Thread Count for Parallel Background Processes](#) for more information.
- **Batch thread count.** Thread count is only used for background processes that can be run in multiple parallel threads. It contains the total number of parallel threads that have been scheduled. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to [Optimal Thread Count for Parallel Background Processes](#) for more information.
- **Batch rerun number.** Rerun number is only used for background processes that download information that belongs to given run number. It should only be supplied if you need to download an historical run (rather than the latest run).
- **Batch business date.** Business date is only used for background processes that use the current date in their processing. For example, billing using the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used. If supplied, this date must be in the format YYYY-MM-DD. Note: this parameter is only used during QA to test how processes behave over time.
- **Override maximum records between commits.** This parameter is optional and overrides each background process's Standard Commit. You would reduce this value, for example, if you were submitting a job during the day and you

wanted more frequent commits to release held resources. You might want to increase this value when a background process is executed at night (or weekends) and you have a lot of memory on your servers.

- **Override maximum minutes between cursor re-initiation.** This parameter is optional and overrides each background process's Standard Cursor Re-Initiation Minutes. You would reduce these values, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources (or more frequent cursor initiations). You might want to increase these values when a background process is executed at night (or weekends) and you have a lot of memory on your servers.
- **User ID.** Please be aware of the following in respect of user ID:
 - The user ID is a user who should have access to all application services in the system. This is because some batch processes update records via services that may check security.
 - Any batch process that stamps a user ID on a record it creates or updates uses this user ID.
 - This user ID's [display profile](#) controls how dates and currency values are formatted in messages.
- **Password.** Password is not currently used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start (Y/N), trace program exit (Y/N), trace SQL (Y/N) and output trace (Y/N).** These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed. If output trace is set to Y, special messages formatted by the background process are written.

NOTE: The information displayed when the output trace switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

Common Additional Parameters

Each batch control supports the definition of additional parameters. There are some additional parameters that are common to all batch processes or common to a specific type of batch process. The batch control should be delivered with the appropriate additional parameters. However, when new additional parameters are introduced, existing batch controls may not be updated with the new additional parameter.

The following table highlights the common parameters that may be linked to a batch control. Note that for batch parameters, although there is a sequence number that controls the displayed order of the parameter, the batch process does not use the sequence to identify a particular parameter but rather uses the parameter name. In some cases multiple parameter names are supported (a 'camel case' version and an 'all caps' version).

Parameter Name	Description	Additional Comments
MAX-ERRORS / maxErrors	Each of the batch processes has, as part of its run parameters, a preset constant that determines how many errors that batch process may encounter before it is required to abort the run. A user can override that constant using this parameter.	The input value must be an integer that is greater than or equal to zero. The maximum valid value for this parameter is 999,999,999,999,999.
DIST-THD-POOL	Each batch process executes in a thread pool. This parameter is only necessary if the batch process should execute in a different thread pool than the default thread pool.	The default thread pool name is DEFAULT .
emailMode	When the batch job is submitted with an associated email address, the default logic is to send an email when the job completes	Valid Values <ul style="list-style-type: none"> • ERROR — send an email only when the job ends in Error status.

Parameter Name	Description	Additional Comments
	regardless of success or failure. Use this parameter to limit the email based on the status of the job when it ends.	<ul style="list-style-type: none"> • SUCCESS — send an email only when the job ends in successfully. • ALL — always send an email only when the job ends. (This is the default.)
The following parameters are only applicable to jobs that use the Thread Level SQL Select method of distributing work to threads as described in Parallel Background Processes .		
overrideLowIdValue	Specifies a new low id to use in calculating the range for a thread. The framework by default assumes that the Id is between 0's (e.g. 000000000) and 9's (e.g. 999999999), but this parameter will override the low value.	The parameter value can be an actual number or it can be set to auto . If auto is configured, it is set to the lowest current value on the database table associated with the background process.
overrideHighIdValue	Specifies a new high id to use in calculating the range for a thread. The framework by default assumes that the Id is between 0's (e.g. 000000000) and 9's (e.g. 999999999), but this parameter will override the high value.	The parameter value can be an actual number or it can be set to auto . If auto is configured, it is set to the highest current value on the database table associated with the background process.
idRangeOverrideClass	Use this parameter to specify a custom class to do thread range calculation. During batch execution, this override class is instantiated and the setter methods called to initialize the Ids as required. The low and high getter methods are called to retrieve the high and low ids to be used for the run.	The class name specified must implement interface <code>com.splwg.base.api.batch.BatchIdRangeOverride</code> .
The following parameters are only applicable to jobs that perform a single commit, for example for extract batch jobs.		
numRecordsToFlush	This parameter defines how frequently to flush the Hibernate cache to prevent high heap consumption and Out Of Memory Errors.	

Specific Batch Parameters

Some background processes define additional parameters that are specific to their functionality. When a process receives additional parameters, they are defined and documented in the batch control entry in the application. They are also visible in the [batch control](#) viewer (part of the [application viewer](#)).

Indicating a File Path

Some of the system background processes use extra parameters to indicate a File Path and/or File Name for an input file or an output file. For example, most extract processes use File Path and File Name parameter to indicate where to place the output file.

When supplying a FILE-PATH variable, the directory specified in the FILE-PATH must already exist and must grant write access to the administrator account for the product. You may need to verify a proper location with your systems administrator.

The syntax of the FILE-PATH depends on the platform used for the product application server. Contact your system administrator for verification. For example, if the platform is UNIX, use forward slashes and be sure to put a trailing slash, for example `/spltemp/filepath/`.

Processing Errors

When a background process detects an error, the error may or may not be related to a specific object that is being processed. For example, if the program finds an error during batch parameter validation, this error is not object-specific. However, if the program finds an error while processing a specific bill, this error is object-specific. The system reports errors in one of the following ways:

- Errors that are not object-specific are written to the error message log in the [Batch Run Tree](#).
- Some batch processes create entries in an "exception table" for certain object-specific errors. For example, an error detected in the creation of a bill in Oracle Utilities Customer Care and Billing may be written to the bill exception table. If an error is written to an exception table, it does not appear in the batch run tree. For each exception table, there is an associated to do entry process that creates a To Do Entry for each error to allow a user to correct the problem on-line.
- For some background processes, errors that do not result in the creation of an exception record may instead generate a To Do entry directly. For these processes, if you wish the system to directly create a To Do entry, you must configure the To Do type appropriately. Refer to [To Do entry for object-specific errors](#) for information about configuring the To Do type. If the background process detects an object specific error AND you have configured the system to create a To Do entry, the error is not written to the batch run tree. If you have configured your To Do type to not create To Do entries for certain errors, these errors are written to the [batch run tree](#).

NOTE: Some processes create exceptions and To Do entries. It is possible for a background process to create entries in an exception table and create To Do entries directly, depending on the error. Consider batch billing in Oracle Utilities Customer Care and Billing; any conditions that cause a bill or bill segment to be created in **error** status result in a record added to the bill exception table or the bill segment exception table. However, any object-specific error that is not related to a specific bill or bill segment or any error that prevents a bill or bill segment from being created may result in a To Do entry for the object-specific error.

Post-Processing Logic

The product supports executing one or more algorithms after all the threads of a given batch job have completed. This allows for some special processing to occur at the end of a batch job. Algorithms for this plug-in spot receive the batch control and run number of the batch job. The following are some examples of functionality that may be executed at the end of a batch job:

- Another dependent batch job can be kicked off. Note that this use case is only needed when the multiple dependent jobs are not part of a scheduler (which can also detect the successful end of one batch job so as to submit the next job).
- Statistics for the batch run may be analyzed and based on results, a To Do Entry may be sent to an administrator.
- If the current batch job is processing a large number of child records in multiple threads, a parent record could be updated to a different status or with some other audit information.

Note that the units of work for all threads are committed prior to executing the post-processing logic. The algorithm should perform standard error handling. If an error occurs in one of the post-processing algorithms, the overall batch job's status is set to Error so that it can be re-submitted to retry the logic in the finalize step.

Timed Batch Processes

Most batch jobs are submitted via a batch scheduler. In the absence of a scheduler, a batch control may be configured as "timed" triggering the framework to monitor and schedule these batch jobs as defined by the timer interval. The timer interval defines the desired interval between starts (in seconds). The system schedules new batch runs at each interval if the last instance of the job has completed.

When configuring a batch control as “timed”, other default information must be provided, including the User ID and Language to use for submitting the job and the email address for notification, if desired.

Timed batch controls also include an Active setting, allowing for an implementation to temporarily stop further executions of the batch job (but retain the other timer settings).

Timed jobs are controlled by the default threadpool and not by a scheduler. When the **DEFAULT** threadpoolworker starts it will start executing any job for a Batch Control configured as **Timed** with the Timer Active set to **Yes**. This is whether the batch daemon or batch server is enabled or not.

Monitor Background Processes

In many areas of the system, functionality is driven from business object configuration as a BO driven record progresses through its lifecycle. Refer to [Business Object Lifecycle](#) for details. As part of that functionality, it is possible that a background process, called a [monitor batch process](#), is used to execute functionality for the record. A single program is provided for the BO monitor functionality and parameters are used to limit the records processed by maintenance object and other optional parameters that may further limit the records. The product typically provides at least one monitor batch control for each maintenance object that supports a configurable lifecycle on its business object.

This topic highlights the parameters supported by the monitor batch job. Not all parameters are applicable to all maintenance objects and therefore may not be configured on a give base monitor batch control.

Parameter Name	Description	Comments
maintenanceObject	Maintenance Object	For most base delivered batch controls, this parameter is delivered already populated with the value of the maintenance object value. Note that it is supported to leave this value blank, at which point, the program will determine the maintenance object (objects) to process by looking for an MO that refers to this batch control record as an option.
isRestrictedByBatchCode	Restrict by Batch Code	Set this to true to indicate whether the process should only select records that explicitly refer to this batch control on its current BO state. This is also referred to as “deferred” mode. If set to false , the program includes all records that refer to the current batch control in its BO state and records that don’t refer to any batch control in its current state (but monitor algorithms exist in the current state). This is commonly referred to as “periodic” mode. Note that if the value is not set at all, the program will determine whether to run it as “deferred” or “periodic” based on whether the batch code is configured on the MO option as a State Monitor Process (“deferred”) or a Periodic Monitor Process .
restrictToType	Restrict by Related Type	This parameter is only applicable to maintenance objects that have a related ‘type’ object and the maintenance object has configured an option indicating the field for the related type column. This parameter may be used to limit the processing to records that are in the indicated type.
restrictToBusinessObject	Restrict by Business Object	This parameter may be used to limit the processing to records that are in the indicated business object.
restrictToStatus	Restrict by Status	This parameter may be used to limit the processing to records that are in the indicated status.
sampleRecordNumber	Sample Record Number	This is not a commonly used parameter. It is only applicable when the monitor is used for a business use case that supports processing a

Parameter Name	Description	Comments
		subset of the records during a testing phase. For example, if the process is validating a large number of records, it may be an option to only validate every 100 records to determine if there are repeated validation errors that may indicate a common problem that may be solved to fix many errors.

Also note that when submitting a monitor process with multiple [parallel threads](#), the program will use a **Thread Level SQL Select** strategy unless any of the following are true (in which case it will use the **Job Level SQL Select** strategy):

- The input maintenance object is left blank and the program finds more than one maintenance object that refers to this batch control in its options.
- A single MO is applicable but it has a multi-part primary key.
- A single MO is applicable and it has a single primary key, but it is a user defined key instead of a system generated key.
- The sample record number parameter is populated.

Plug-in Driven Background Processes

Although the product is delivered with a rich library of background processes, implementations may have business requirements that require new processes to be introduced. It is possible for an implementation to write a background process from scratch using a base process as a template. However, the product also provides base background processes that call algorithms to select the records to be processed and to process the records. These are called plug-in driven background processes. The base processes implement standard background process functionality including parallel background process logic, the ability to create To Do entries for errors. This allows for an implementation to take advantage of the pre-built support and provide plug-ins that include the logic that is unique to the specific use case.

The following sections provide more information about the provided functionality.

Types of Processes

The system provides the following processes that support plug-ins for selecting and processing the records:

- Ad-hoc Process. This background process is provided for implementations that have some custom business logic that needs to be performed on a group of records. The base batch control Plug-in Driven Generic Template ([F1-PDBG](#)) may be used as a template.
- Extract Process. This background process is provided for implementations that have extract files to produce for integration with external systems. The process includes parameters to configure the file path and file name for the created file along with other parameters to control how the file is formatted. The base batch control Plug-in Driven Extract Template ([F1-PDBEX](#)) may be used as a template.

Select Records Algorithm

The first important algorithm to design when implementing a plug-in driven batch process is the Select Records algorithm, plugged in on the [batch control](#) page. This algorithm type must define the first parameter as the SQL. The batch job will directly access the SQL parameter value in the metadata (rather than invoking the algorithm). All other parameters are available for the algorithm to use for its own logic.

In addition, when invoking the algorithm, it must return the strategy to use (**Thread Level SQL Select** or **Job Level SQL Select**). Refer to [parallel background processes](#) for more information about the two strategies and when to use each. When choosing the **Thread Level SQL Select** strategy, the algorithm should return the name of the primary key in the Key Field parameter. In addition, the SQL should include a **BETWEEN** clause that includes the bind variables for the low and high ID for the ranges. See below for the bind variable syntax.

If the SQL statement includes variables that are determined at execution time, it must use bind parameters. Bind parameters are referenced in the SQL statement using a colon and a parameter name (for example **:parameter**). There are some variables provided by the system that are populated by the batch job at execution time. These have **fl.** as its prefix.

The system supports the following pre-defined bind parameters:

- **:fl.lowID** and **:fl.highID** - these should be used in the **BETWEEN** clause for the **Thread Level SQL Select** strategy. The batch job will substitute the appropriate ID range as required.
- **:fl.batchCode** and **:fl.batchNumber** - these are common attributes of the batch control that are referenced on a record for selection purposes. Note that the batch run number is set according to whether the batch job is a re-run of a previous run or not.
- **:fl.businessDate** - the batch job will populate the input batch business date, if populated otherwise the current date.

For any other custom parameters, the Select Records algorithm may return one or more name / value pairs where the name matches a bind variable in the SQL. Note that the name cannot start with **fl.** as its prefix. The batch job will use the value returned by the algorithm to set the bind parameter in the SQL statement.

The product provides a base algorithm type for this plug-in spot that simply defines a parameter for the SQL. It also includes parameters for the strategy and the key field name. This algorithm type may be used by any custom batch process where the SQL does not rely on any special bind variables that must be determined. Simply create an algorithm for the algorithm type and provide the appropriate SQL. Refer to the algorithm type Select Records by Predefined Query ([F1-PDB-SR](#)) for more information.

Process Records Algorithm

The other important algorithm to design when implementing a plug-in driven batch process is the Process Record algorithm, plugged in on the [batch control](#) page. This algorithm is called for each record selected for the process. It receives all the information that was selected from the Select Records plug-in.

For the ad-hoc processing batch process, algorithms plugged into this spot are responsible for doing the work for each record based on the desired logic.

For the extract batch process, algorithms plugged into this spot are responsible for returning the data that should be written to the file in one or more XML instances along with the schema name(s) that describes the XML instance(s). For XML output format, the batch process will write the XML instance data as returned by the plug-in. For fixed position or CSV output format, the batch process will convert the XML instance data to the appropriate format and add it to the file.

Also note that algorithms for this plug-in spot will be passed two Booleans, **isFirst** and **isLast**, to indicate if the current work unit is the first and/or last for that thread. This allows for the plug-in to do additional work if needed. Note that the **isFirst** indication is available for both types of batch processes, ad-hoc and extract. However, the **isLast** indication is only applicable for the file extract batch. For the ad-hoc batch process this value will always be set to **false**. Extracts will always execute in a single database transaction. In a single transaction run, any error causes the run to be aborted so that it restarts from the beginning when resubmitted. This is done to avoid partial files being written along with inaccurate setting of the **isLast** element.

The product provides a base algorithm type for this plug-in spot that illustrates the technique to follow when implementing an extract type of plug-in driven background process. Refer to the algorithm type General Process - Sample Process Record Extract ([F1-GENPROCEX](#)) for more information.

Configuring a New Process

The following points summarize the steps needed to implement a new background process using plug-ins for the specific functionality:

- Verify the SQL that the background process should execute. Keep in mind that all the data selected in the SQL is available to pass to the plug-in that processes the records. If the performance of the background process is important, be sure to consult with a DBA when designing the SQL.

- If the SQL does not require any custom variables to substitute at runtime, create an algorithm for the base algorithm type [F1-PDB-SR](#) and configure the SQL. In addition, configure the strategy and the primary key name (for the **Thread Level SQL Select** strategy).
- If the SQL requires custom variables, a new [plug-in script](#) must be designed and coded to populate the variable names and values using the algorithm entity **Batch Control — Select Records**. Besides defining the variables, the algorithm must also indicate the strategy and the primary key name (for the **Thread Level SQL Select** strategy). Define the algorithm type for the newly created script. The first parameter of the algorithm type must be the SQL as illustrated in the base algorithm type. Note that the other parameters are available for use by this algorithm type if needed. Define the algorithm, populating the SQL as appropriate (using the custom variables).
- Design the logic required for each record and create a [plug-in script](#) where the algorithm entity is **Batch Control — Process Record**. Note that the plug-in receives all the information selected in the SQL defined in the Select Records plug-in
 - For an ad-hoc process, the algorithm should perform whatever process is required based on the business use case. Note that the background process is responsible for committing the records.
 - For an extract process, the algorithm is responsible for returning one or more schema instances populated with information that should be written to the file. If an existing schema satisfies the output requirements, it may be used. Otherwise, define a data area to indicate the output format of the records as appropriate.

In either case, define the algorithm type and algorithm for the newly created script.

- Create a batch control by duplicating the appropriate base template as per the type of background process needed. Plug in the algorithms created above and configure the parameters as appropriate.

How to Re-extract Information

If you need to recreate the records associated with an historical execution of an extract process, you can - simply supply the desired batch number when you request the batch process.

How to Submit Batch Jobs

Most batch jobs are submitted via a batch scheduler. Refer to [Batch Scheduler Integration](#) for information about the integration with the Oracle Scheduler.

Batch jobs may be configured as [Timed](#), which means they will automatically be run based on the timer frequency.

In addition, you can manually submit your adhoc background processes or submit a special run for one of your scheduled background processes using the online [batch job submission](#) page.

How to Track Batch Jobs

You can track batch jobs using the batch process pages, which show the execution status of batch processes. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run.

FASTPATH: For more information, refer to [Tracking Batch Processes](#).

How to Restart Failed Jobs and Processes

Every process in the system can be easily restarted if it fails (after fixing the cause of the failure). All you have to do is resubmit the failed job; the system handles the restart.

Assessing Level of Service

For some background processes, an implementation may wish to supply an algorithm that checks some conditions to assess whether or not the process is performing as expected. This algorithm could be simply verifying that the batch process is running as frequently as expected. Or it could be used to check the performance of the job to see if it is running as efficiently as expected. Or it could analyze the data processed by the background process to assess whether there may be some problem with the quality of the data.

The system provides a Level of Service plug-in spot on [batch control](#) to configure the appropriate algorithm for a given background process, if desired. The algorithm is expected to return a value to indicate the 'level of service' determined along with a message indicating the reason for the value. The following Level of Service values are supported:

- **Normal.** Indicates that the algorithm did not detect any issues.
- **Warning.** Indicates that the algorithm found some issues that may or may not indicate a problem.
- **Error.** Indicates that the algorithm found some issues that should be investigated.
- **Disabled.** Indicates that the algorithm could not properly execute the level of service logic.

When viewing a [batch control](#) record, if there is a level of service algorithm configured, its logic is executed and the results are displayed. The level of service is also part of the [Health Check](#) service.

System Background Processes

NOTE: List of system background processes. The list of background processes provided in the base product may be viewed in the [application viewer's batch control](#) viewer. In addition if your implementation adds batch control records, you may [regenerate](#) the application viewer to see your additions reflected there.

Defining Batch Controls

The system is delivered with all necessary batch controls. Implementations may define default values for parameters. In addition, implementations may define their own background processes.

To view background processes, open **Admin > System > Batch Control**. Refer to [Background Processing Concepts](#) for more information.

CAUTION: Important! If you introduce a new batch process, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Description of Page

Enter an easily recognizable **Batch Process** and **Description** for each batch process.

Owner indicates if this batch control is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a batch control. This information is display-only.

Use the **Detailed Description** to describe the functionality of the batch process in detail.

Use **Batch Control Type** to define the batch process as either **Timed** or **Not Timed**. A [Timed batch process](#) will be automatically initialized on a regular basis. A Not Timed process needs to be run manually or through a scheduler.

Use **Batch Control Category** to categorize the process for documentation purposes. The base values provided are as follows:

- **Ad Hoc.** Processes of this type are run on an ad hoc basis, only when needed. For example, if there is a process to do a mass cancel / correction of data, it would only be run when a situation occurs requiring this.

- **Extract.** Extract processes extract information that is interfaced out of the system. Processes of this type typically extract records marked with a given run number. If the requester of the process does not supply a specific run number, the system assumes that the latest run number should be extracted. If you need to re-extract an historical batch, you simply supply the respective run number when you request the batch process.
- **ILM. Information Lifecycle Management** jobs are crawler background processes that are associated with the ILM based storage solution.
- **Monitor.** Processes of this type are processes related to business objects with a lifecycle state that defines [monitor algorithms](#). The monitor process selects records in a given state and executes its algorithms, which may cause the record to transition to another state or may trigger some other logic to occur. Using configuration, the monitor process may target only specific records. Refer to [Monitoring Batch Processes](#) for more information. Note that these types of background processes can be considered a subset of **Process What's Ready**
- **Process What's Ready.** Processes of this type create and update records that are “ready” for processing. The definition of “ready” differs for every process. For example, a payment upload process creates payments for every record that is **pending**. An overdue event monitor activates pending overdue events that have reached their trigger date.
- **Purge.** Processes of this type are used to purge historical records from certain objects that generate a large number of entries and may become unwieldy over time.
- **To Do Entry.** Processes of this type are used to detect a given situation and create or complete a To Do Entry. Refer to [To Do Entries Created by Background Processes](#) for more information.
- The following categories are related to the data conversion / migration processes, which are not applicable to all products:
 - **Conversion.** Processes of this type are dedicated to converting or migrating data from external applications into the product.
 - **Object Validation.** Processes of this type are dedicated to validate data within objects for conversion or migration purposes.
 - **Referential Integrity.** Processes of this type are dedicated to validate referential integrity within objects for conversion or migration purposes.

NOTE: Additional categories may be introduced by your specific product.

If the batch process is Timed, then the following fields are available:

- **Timer Interval** is the number of seconds between batch process submissions. The system will start the next run this many seconds from the start time of the previous run.
- **User ID** is the ID under which the batch process will run.
- **Email Address** is the Email address to be used for notification if the batch process fails.
- **Timer Active** allows you to temporarily switch off the timer while retaining the other settings for the timed job.
- **Batch Language** is the language associated with the batch process.

Use **Program Type** to define if the batch process program is written in **Java** or **Java (Converted)**, meaning that the program has been converted into Java.

NOTE: **Java (Converted)** program types are not applicable to all products.

Use **Program Name** to define the Java class / program associated with your batch process:

NOTE: View the source. If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#).

Level of Service shows the output of the [level of service](#) algorithm for the batch control. It shows the level of service lookup value along with a message indicating the reason for the output value. Note that if no level of service algorithm is found, then the value **Disabled** is shown with a message indicating that no algorithm is provided for this batch control.

The **Last Update Timestamp**, **Last Update Instance** and **Next Batch Nbr** are used for audit purposes.

Turn on **Accumulate All Instances** to control how this batch control is displayed in the [Batch Run Tree](#). If checked, the run statistics (i.e., "Records Processed" and "Records in Error") for a thread are to be accumulated from all the instances of the thread. This would include the original thread instance, as well as any restarted instances. If this is not turned on, only the ending (last) thread instance's statistics are used as the thread's statistics. This may be preferred for certain types of batch processes where the accumulation would create inaccurate thread statistics, such as those that process flat files and, therefore, always start at the beginning, even in the case of a restart.

The following fields are default values that are used when a batch job is submitted for the batch control:

- Use **Thread Count** to control whether a background process is run single threaded or in multiple parallel threads. This value defines the total number of threads that have been scheduled.
- Select **Trace Program Start** if you want a message to be written whenever a program is started.
- Select **Trace SQL** if you want a message to be written whenever an SQL statement is executed.
- Use **Override Nbr Records to Commit** to define the default number of records to commit. This is used as the default value for timed jobs as well as online submission of jobs that are not timed.
- Select **Trace Program Exit** if you want a message to be written whenever a program is exited.
- Select **Trace Output** if you want a message to be displayed for special information logged by the background process.

For more information about these fields, see [Batch Job Submission - Main](#)

The parameter collection is used to define additional parameters required for a particular background process. The following fields should be defined for each parameter:

Sequence. Defines the relative position of the parameter.

Parameter Name. The name of the parameter as defined by the background process program.

Description. A description of the parameter.

Detailed Description. A more detailed description of the parameter.

Required. Indicates whether or not this is a required parameter.

Parameter Value. The default value, if applicable. Note that an implementation may define a default value for base provided batch controls.

Security. Indicates whether the system should **Encrypt** the parameter value or not. A value of **Encrypt** means that the parameter value is stored in the database and written to the log files using encryption. In addition, the parameter is written to the log files with asterisks. The setting applies to values entered here as well as in the online [Batch Submission](#). If there is no need to secure the parameter value, use the default setting of **None**.

Owner Indicates if this batch process is owned by the base package or by your implementation (Customer Modification). The system sets the owner to **Customer Modification** when you add a batch process. This information is display-only.

Batch Control - Algorithms

Use this page to maintain a batch control's algorithms. Open this page using **Admin > System > Batch Control** and then navigate to the **Algorithms** tab.

Description of Page

The **Algorithms** grid contains algorithms that control important functions for instances of this batch control. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes the **System Events**.

System Event	Optional / Required	Description
Level of Service	Optional	Algorithms of this type are called to determine the Level of Service provided by a batch control. Note that only one algorithm is allowed. Refer to Assessing Level of Service for more information. Click here to see the algorithm types available for this system event.
Post-Processing	Optional	Algorithms of this type are called after all threads are complete. Multiple algorithms are allowed and are executed in sequence order. Refer to Post-Processing Logic for more information. Click here to see the algorithm types available for this system event.
Process Record	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to process a specific record. Click here to see the algorithm types available for this system event.
Select Records	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to define the SQL to use to select the records to process. Only one algorithm is allowed. Click here to see the algorithm types available for this system event.

NOTE: You can add new system events. Your implementation may want to add additional batch control oriented system events. To do that, add your new values to the customizable lookup field **F1_BATCH_CTRL_SEVT_FLG**.

On-Line Batch Submission

The on-line [batch submission](#) page enables you to request a specific background process to be run. When submitting a background process on-line, you may override standard system parameters and you may be required to supply additional parameters for your specific background process. After submitting your background process, you may use this page to review the status of the submission.

The following topics further describe logic available for on-line submission of background processes.

Batch Submission Creates a Batch Run

When you request a batch job to be submitted from on-line, the execution of the desired background process will result in the creation of a batch run. Just as with background processes executed through your scheduler, you may use the [Batch Run Tree](#) page to view the status of the run, the status of each thread, the run-instances of each thread, and any messages that might have occurred during the run.

NOTE: Your on-line submission record is assigned a status value so that you may know whether your job has been submitted and whether or not it has ended, however, it will not contain any information about the results of the background process itself. You must navigate to the [Batch Run Tree](#) page to view this detail.

Jobs Submitted in the Background

When you save a record on the batch job submission page, the batch job does not get submitted automatically. Rather, it saves a record in the batch job table. A special background process will periodically check this table for pending records and will execute the batch job. This background process will update the status of the batch job submission record so that a user can determine when their job is complete.

NOTE: At installation time, your system administrator will set up this special background process to periodically check for pending records in the batch job submission table. Your administrator will define how often the system will look for pending records in this table.

It should be noted that this special background process only submits one pending batch job submission record at a time. It submits a job and waits for it to end before submitting the next pending job.

NOTE: If you request a batch job to be run multi-threaded, the special background process will submit the job as requested. It will wait for all threads to complete before marking the batch job submission record as **ended**. Refer to [Running Multi-threaded Processes](#) for more information.

Email Notification

If you wish the system to inform you when the background process completes, you may supply your email address. The email you receive will contain details related to the batch job's output; similar to the job results you would see from your batch scheduler.

NOTE: This assumes that during the installation process, your system administrator configured the system to enable email notification. Your administrator may also override the amount of detail included in the email notification.

Running Multi-Threaded Processes

Many of the system background processes may be run multi-threaded. When submitting a background process on-line, you may also run a multi-threaded process or run a single thread of a multi-threaded process. The fields Thread Count and Thread Number on the [batch submission](#) page control the multi-threaded process requests:

- To run a multi-threaded process, indicate the number of threads in **Thread Count** and enter **0** in the **Thread Number**.
- To run a single thread in a multi-threaded process, indicate the number of threads in Thread Count and indicate the Thread Number you would like to run.
- To run a process as a single thread, enter Thread Count = **1** and Thread Number = **1**. This will execute the background process single-threaded.

NOTE: When running a multi-threaded process, the special background process will wait until all threads are complete before marking the batch job submission record as **Ended**.

Batch Jobs May End in Error

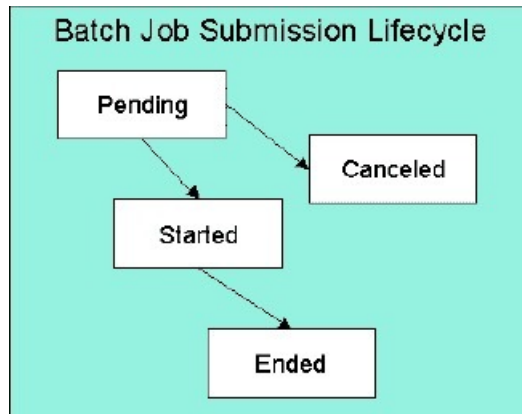
It is possible for your background process to end with an error. When this occurs, your batch job submission record will still be marked as **Ended**. You will need to navigate to the [Batch Run Tree](#) page to determine the status of the batch run.

Submitting Jobs in the Future

If you wish to request a batch job to be submitted in the future, you may do so when creating your batch job submission record by entering a future submission date. The special background process, which looks for pending records in the batch job submission table, will only submit batch jobs that do not have a future submission date.

Lifecycle of a Batch Job Submission

The following diagram illustrates the lifecycle of a batch job submission record.



Pending — Records are created in **Pending** status. Records in this state are put in a queue to be submitted.

Canceled— Users may cancel a pending record to prevent the batch job from being submitted.

Started— Once a pending record has been submitted for processing, its status will be changed to **Started**. Records in this status may not be canceled.

Ended— When the batch job has finished processing, its status will be changed to **Ended**. Note that records in **Ended** status may have ended in error. Refer to [Batch Jobs May End in Error](#) for more information.

Batch Job Submission - Main

This page allows you to submit a batch job on-line. Navigate to this page using **Menu > Tools > Batch Job Submission**.

Description of Page

The **Batch Job ID** is a system generated random number that identifies a particular submission.

To submit a batch job, choose the **Batch Code** for the process you wish to submit.

NOTE: List of system background processes. The list of background processes provided in the base product may be viewed in the [application viewer's batch control](#) viewer.

The following parameters are provided with each background process:

Thread Number is used to control whether a background processes is run single threaded or in multiple parallel threads. It contains the relative thread number of the process. For example, if the process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20). Refer to [Running Multi-threaded Processes](#) for more information about populating this field.

NOTE: Not all processes may be run multi-threaded. Refer to the description of a batch control to find out if it runs multi-threaded.

Thread Count is used to control whether a background processes is run single threaded or in multiple parallel threads. It contains the total number of threads that have been scheduled. For example, if the process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to [Running Multi-threaded Processes](#) for more information about populating this field.

Batch Rerun Number is only used for background processes that download information that belongs to given run number. It should only be supplied if you need to download an historical run (rather than the latest run).

Batch Business Date is only used for background processes that use a date in their processing. In Oracle Utilities Customer Care and Billing, for example, a billing process can use the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used at the time the background process is executed.

NOTE: Saving a record on this page does not submit the batch job immediately. A special background process will run periodically to find pending records and submit them. Depending on how often the special process checks for pending records and depending on how many other pending records are in the 'queue', there may be a slight lag in submission time. If the desired execution date/time is close to midnight, it is possible that your batch job will run on the day after you submit it. If you have left the business date blank in this case, keep in mind that your business date would be set to the day after you submit the job.

Override Nbr Records To Commit and **Override Max Timeout Minutes**. These parameters are optional and override each background process's Standard Commit Records and Standard Cursor Re-Initiation Minutes. (Each background process's Standard Commit Records / Standard Cursor Re-Initiation Minutes should be documented in the detailed description of the batch control record). Note that Max Timeout Minutes corresponds to the Cursor Re-initiation Minutes.

FASTPATH: Refer to [Parameters Supplied to Background Processes](#) for more information.

User ID is the user ID associated with the run of the background process. Refer to [Parameters Supplied to Background Processes](#) for more information about the significance of the user id.

NOTE: This field defaults to the id of the current user.

Password is not currently used.

Language Code is used to access language-specific control table values. For example, error messages are presented in this language code.

If you wish the system to notify you when the batch job is complete, enter your **Email ID**. Refer to [Email Notification](#) for more information.

NOTE: This field defaults to the email address for the current user, if populated on the [user](#) record.

The **Desired Execution Date/Time** defaults to the current date and time. Override this information if you wish the background process to be executed at some future date and time. Refer to [Submitting Jobs in the Future](#) for more information.

The **Batch Job Status** indicates the current status of the batch job. Refer to [Lifecycle of a Batch Job Submission](#) for more information.

The **Program Name** associated with the batch control code is displayed.

The following trace parameters may also be supplied to a background process and are only used during QA and benchmarking.

- **Trace Program Start** Turn on this switch if you wish a message to be written whenever a program is started.
- **Trace Program Exit** Turn on this switch if you wish a message to be written whenever a program is exited.

- **Trace SQL** Turn on this switch if you wish a message to be written whenever an SQL statement is executed.
- **Trace Output** Turn on this switch if you wish a message to be displayed for special information logged by the background process.

NOTE: The information displayed when the trace output switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

NOTE: The location of the output of this trace information is defined by your system administrator at installation time.

If additional parameters have been defined for this background process on the Batch Control page, the **Parameter Name**, **Description**, **Detailed Description** and an indicator of whether or not the parameter is **Required** are displayed.

If a default parameter value is configured on the batch control configuration, that value is shown and may be overridden. Confirm or enter the appropriate **Parameter Value** for each parameter. Note that if the parameter value is configured to be Encrypted on the batch control configuration, the value here will be shown encrypted.

Once you have entered all the desired values, **Save** the record in order to include it in the queue for background processes.

If you wish to duplicate an existing batch job submission record, including all its parameter settings, display the submission record you wish to duplicate and use the **Duplicate and Queue** button. This will create a new Batch Job Submission entry in pending status. The new submission entry will be displayed.

If you wish to cancel a **Pending** batch job submission record use the **Cancel** button. The button is disabled for all other status values.

Tracking Batch Processes

The batch process pages show the execution status of batch processes. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run. Refer to [Defining Batch Controls](#) for more information on how batch control codes are defined.

Batch Run Tree - Main

This page allows you to view the status of a specific execution of a batch job. Navigate to this page using **Menu > Tools > Batch Run Tree**.

Description of Page

Select a **Batch Control** process and **Batch Number** to view information and statistics on the batch run's "threads". The following points should help understand this concept:

- Many batch jobs cannot take advantage of your hardware's processing power when they are run singularly. Rather, you'll find that a large percentage of the CPU and/or disk drives are idle.
- In order to minimize the amount of idle time (and increase the throughput of your batch processes), we allow you to set up your batch processes so that multiple instances of a given batch job are executed at the same time. For example, in Oracle Utilities Customer Care and Billing when you schedule the **billing** process, you can indicate that multiple parallel instances should be executed (rather than just one instance). You'd do this so that the processing burden of creating bills for your customers can be spread over multiple processes.
- We refer to each parallel execution of a batch process as a "thread".
- Statistics and information messages are displayed in respect of each thread. Why? Because each thread is a separate execution and therefore can start and end at different times.

The tree includes a node that displays the total number of records processed for the batch run, the total number of records in error for the batch run and the batch run elapsed time. The elapsed time is the longest elapsed time among the batch thread(s). The message is red if there are any records in error.

If the background process has been enabled to create [To Do entries for object specific errors](#), information about the To Do entries are displayed in the tree. This information is not displayed for each thread, but rather all the To Do entries created for the batch run are grouped together. The To Do entries are grouped by their status.

If the application properties file has been configured with the location of the log files and the log files associated with the batch thread are still available, the links **Download stdout** and (if applicable) **Download stderr** are visible. Clicking either link allows you to view or save the log files.

NOTE: Security Access. The 'download' hyperlinks are only visible for users that have security access to the **Download** access mode for the batch run tree application service.

The messages that appear under a thread always show the start and end times of the execution instance. If errors are detected during the execution of the thread, these error messages may also appear in the tree. Refer to [Processing Errors](#) for information about the types of errors that appear in the batch run tree.

Batch Run Tree - Run Control

By default, if a batch process fails, it will restart. This tab allows you to modify the restart status of a failed run.

Navigate to this page using **Menu > Tools > Batch Run Tree** search for the desired batch control and then navigate to the **Run Control** page.

Description of Page

On the main page, you must select a **Batch Control**, **Batch Number**, and **Batch Rerun Number** to view a tree of the batch run. On this page, the following information is displayed:

- **Date Time** contains the date and time this batch run last start or last completed.
- **Batch Business Date** is the business date that was supplied to the background process (this date is used as the "system date" by the process).

Run Status indicates the status of the batch run. Valid values are: **In Progress**, **Error**, and **Complete**.

If the **Run Status** is **Error**, the system will attempt to restart this run when you attempt to execute the **Batch Control**. In most situations, this is exactly what you want to happen. However, there are rare situations where you do not want the system to execute a given batch run (e.g., if this run is somehow corrupt and you cannot correct the data for whatever reasons). If you want the system to skip the execution of a batch run (and proceed to the next run), turn on **Do Not Attempt Restart**.

Service Health Check

The system provides a service that returns an overall assessment of the system's health. The overall response is expressed using an HTTP code. The codes supported by the service are defined in the lookup **HEALTH_RESPONSE_FLG** and are as follows:

- A value of 500 (One or More Critical Functions Degraded) is returned if there is any critical issue detected by the service.
- A value of 203 (Non-Critical Function Degraded) is returned if no critical issues are found and there is at least one non-critical issue detected by the service.
- Otherwise a value of 200 (All Checks Successful) is returned.

This health check service is accessible through the business service **F1-HealthCheck**. Also note that the system provides an [Inbound Web Service](#) for this business service (also called **F1-HealthCheck**) allowing external systems to use a web service to retrieve this information.

In addition, the product provides a portal that allows a user to view the detailed results of the health check service.

Navigate using **Admin > System > Health Check** to view this portal.

The zone on the portal displays the following:

Overall Response is the HTTP response code returned by the business service as described above.

The grid displays the detail of each individual component checked as part of the system health check. See below for more detail about what components are checked.

- **Health Component** indicate the type of health check performed. Currently the system supports **Batch Level of Service** component type.
- **Health Component Details** provides information about the individual entity checked. The information displayed here depends on the type of health component. If supported by the type of component, the column may be hypertext, allowing the user to drill into the entity itself.
- **Status** displays the status returned by the health component check for this individual entity. The information displayed here depends on the type of health component.
- **Status Reason** provides more detail about why the individual entity is in the indicated status. The information displayed here depends on the type of health component.
- **Response** shows the health check response code assigned to this individual entity's health check response. It is mapped based on the status returned by the health component check.

Health Component Type Details

The details returned by the business service for this portal depends on the health component type. The system supports the **Batch Level of Service** health component type.

This health component type finds all the batch controls that are configured with a [level of service](#) algorithm and invokes the algorithm for each batch control. The business service populates the output for this health service for each batch control as follows:

- The **Health Component Detail** is populated with the Batch Control code and description. In addition, the navigation information for being able to drill into the batch control are provided and used to build the column as hypertext.
- The **Status** is populated with the description of the Level of Service lookup value returned by the level of service algorithm for this batch control.
- The **Status Reason** is populated with the expanded text of the status reason returned by the level of service algorithm for this batch control.
- The **Response** is populated based on the value of the Level of Service status. It is set to **All Checks Successful** (200) when the Level of Service is **Normal** or **Disabled**; **Non-Critical Function Degraded** (203) when the Level of Service is **Warning** and **One or More Critical Functions Degraded** (500) when the Level of Service is **Error**.

The Big Picture of Requests

Requests enable an implementer to design an ad-hoc batch process using the configuration tools.

An example of such a process might be to send an email to a group of users that summarizes the To Do entries that are assigned to them. This is just one example. The request enables many types of diverse processing.

Request Type Defines Parameters

For each type of process that your implementation wants, you must configure a request type to capture the appropriate parameters needed by the process.

Previewing and Submitting a Request

To submit a new request, go to **Menu > Tools > Request** in add mode. You must select the appropriate request type and then enter the desired parameter values, if applicable.

After entering the parameters, the following actions are possible:

- Click **Save** to submit the request.
- Click **Cancel** to cancel the request.
- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can click **Save** to submit the request, **Back** to adjust the parameters, or **Cancel** to cancel the request.

When a request is saved, the job is not immediately submitted for real time processing. The record is saved with the status **Pending** and a monitor process for this record's business object is responsible for transitioning the record to **Complete**.

As long as the record is still **Pending**, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

The actual work of the request, such as generating an email, is performed when transitioning to **Complete** (using an enter processing algorithm for the business object).

To Do Summary Email

The base product includes a sample request process that sends an email to users that have incomplete To Dos older than a specified number of days.

The following configuration tasks are required to use this process:

- Define an Outbound Message Type. The business object usually defined for the Outbound Message Type is **F1-EmailMessage**.
- Define an External System that contains the Outbound Message Type in one of its steps. In the configuration determine if the communication is through SOA, batch, or real-time processing method when sending the email notification. Refer to [Outbound Messages](#) for more information about configuration needed for the different processing methods.
- Create a Request Type that includes the Outbound Message Type and the corresponding External System.
- Create a Request for the created Request Type.

Exploring Request Data Relationships

Use the following links to open the application viewer where you can explore the physical tables and data relationships behind the request functionality:

- Click [F1-REQ-TYPE](#) to view the request type maintenance object's tables.
- Click [F1-REQ](#) to view the request maintenance object's tables.

Defining a New Request

To design a new ad-hoc batch job that users can submit via Request, first create a new Request Type business object. The base product BO for request type, **F1-TodoSumEmailTyp**, may be used as a sample.

The business object for the request includes the functionality for selecting the records to process, displaying a preview map for the user to review, and for performing the actual processing. The base product BO for request, **F1-TodoSumEmailReq**, may be used as a sample. The following points highlight the important configuration details for this business object:

- Special BO options are available for request BOs to support the Preview Request functionality.
 - **Request Preview Service Script.** This script retrieves the information that is displayed when a user asks for a preview of a request.
 - **Request Preview Map.** This map displays the preview of a request.
- The enter algorithm plugged into the Complete state is responsible for selecting the records that satisfy the criteria and processing the records accordingly.

Setting Up Request Types

Use the Request Type portal to define the parameters to capture when submitting a request. Open this page using **Admin > General > Request Type**.

This topic describes the base-package zones that appear on the Request Type portal.

Request Type List. The Request Type List zone lists every request type. The following functions are available:

- Click a **broadcast** icon to open other zones that contain more information about the request type.
- Click **Add** in the zone's title bar to add a new request type.

Request Type. The Request Type zone contains display-only information about a request type. This zone appears when a request type has been broadcast from the Request Type List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click **Edit** to start a business process that updates the request type.
- Click **Delete** to start a business process that deletes the request type.
- Click **Deactivate** start a business process that deactivates the request type.
- Click **Duplicate** to start a business process that duplicates the request type.
- State transition buttons are available to transition the request type to an appropriate next state.

Please see the zone's help text for information about this zone's fields.

Maintaining Requests

Use the Request transaction to view and maintain pending or historic requests.

Open this page using **Menu > Tools > Request > Search**. This topic describes the base-package portals and zones on this page.

Request Query. Use the query portal to search for an existing request. Once a request is selected, you are brought to the maintenance portal to view and maintain the selected record.

Request Portal. This portal appears when a request has been selected from the Request Query portal. The following base-package zones appear on this portal:

- **Actions.** This is a standard actions zone.
- **Request.** The Request zone contains display-only information about a request. Please see the zone's help text for information about this zone's fields.
- **Request Log.** This is a standard log zone.

Defining Algorithms

In this section, we describe how to set up the user-defined algorithms that perform many important functions including:

- Validating the format of a phone number entered by a user.
- Validating the format of a latitude/longitude geographic code entered by a user.
- In products that support payment and billing:
 - Calculating late payment charges.
 - Calculating the recommended deposit amount.
 - Constructing your GL account during the interface of financial transactions to your GL
- And many other functions...

The Big Picture Of Algorithms

Many functions in the system are performed using a user-defined algorithm. For example, when a CSR requests a customer's recommended deposit amount, the system calls the deposit recommendation algorithm. This algorithm calculates the recommended deposit amount and returns it to the caller.

NOTE: Algorithm = Plug-in. We use the terms plug-in and algorithm interchangeably throughout this documentation.

So how does the system know which algorithm to call? When you set up the system's control tables, you define which algorithm to use for each component-driven function. You do this on the control table that governs each respective function. For example:

- You define the algorithm used to validate a phone number on your phone types.
- You define the algorithm in Oracle Utilities Customer Care and Billing used to calculate late payment charges on each Service Agreement Type that has late payment charges.
- The list goes on...

The topics in this section provide background information about a variety of algorithm issues.

Algorithm Type Versus Algorithm

You have to differentiate between the type of algorithm and the algorithm itself.

- An **Algorithm Type** defines the program that is called when an algorithm of this type is executed. It also defines the types of parameters that must be supplied to algorithms of this type.
- An **Algorithm** references an **Algorithm Type**. It also defines the value of each parameter. It is the algorithm that is referenced on the various control tables.

FASTPATH: Refer to [How to Add A New Algorithm](#) for an example that will further clarify the difference between an algorithm and an algorithm type.

How To Add A New Algorithm

Before you can add a new algorithm, you must determine if you can use one of the sample algorithm types supplied with the system. Refer to [List of Algorithm Types](#) for a complete list of algorithm types.

If you can use one of the sample algorithm types, simply add the algorithm and then reference it on the respective control table. Refer to [Setting Up Algorithms](#) for how to do this.

If you have to add a new algorithm type, you may have to involve a programmer. Let's use an example to help clarify what you can do versus what a programmer must do. Assume that you require an additional geographic type validation algorithm. To create your own algorithm type you must:

- Write a new program to validate geographic type in the appropriate manner. Alternatively, you may configure a plug-in script to implement the validation rules. The advantage of the latter is that it does not require programming. Refer to [plug-in script](#) for more information.
- Create an Algorithm Type called **Our Geographic Format** (or something appropriate). On this algorithm type, you'd define the name of the program (or the plug-in script) that performs the function. You'd also define the various parameters required by this type of algorithm.
- After creating the new Algorithm Type, you can reference it on an Algorithm.
- And finally, you'd reference the new Algorithm on the Geographic Type that requires this validation.

Minimizing The Impact Of Future Upgrades

The system has been designed to use algorithms so an implementation can introduce their own logic in a way that's 100% upgradeable (without the need to retrofit logic). The following points describe strong recommendations about how to construct new algorithm type programs so that you won't have to make program changes during future upgrades:

- Do not alter an algorithm type's hard parameters. For example, you might be tempted to redefine or initialize parameters defined in an algorithm type's linkage section. Do not do this.
- Follow the naming conventions for the new algorithm type code and your source code, i.e., both the source code and the algorithm type should be prefixed with "CM". The reason for this naming convention is to make it impossible for a new, base-package algorithm type from overwriting your source code or algorithm type meta-data (we will never develop a program or introduce meta-data beginning with CM).
- Avoid using inline SQL to perform insert/update/delete. Rather, invoke the base-package's object routines or common routines.
- Avoid using base messages (outside of common messages, i.e., those with a message number < 1000) as we may deprecate or change these messages in future releases. The most common problem is caused when an implementation clones a base package algorithm type program because they need to change a few lines of logic. Technically, to be 100% upgradeable, you should add new messages in the "90000" or greater category (i.e., the category reserved for implementation-specific messages) for every message in your new program even though these messages may be duplicates of those in the base package.

Setting Up Algorithm Types

The system provides many algorithm types to support base product functionality. If you need to introduce a new type of algorithm, open **Admin > System > Algorithm Type**.

FASTPATH: Refer to [The Big Picture Of Algorithms](#) for more information.

CAUTION: Important! If you introduce a new algorithm type, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Description of Page

Enter an easily recognizable **Algorithm Type** and **Description**.

Owner indicates if this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type. This information is display-only.

Enter a **Detailed Description** that describes, in detail, what algorithms of this type do.

Use **Algorithm Entity** to define where algorithms of this type can be "plugged in". If a detailed description about an algorithm entity is available, a small help icon is visible adjacent to the dropdown. Click the icon to view the information.

NOTE: The values for this field are customizable using the [lookup](#) table. This field name is **ALG_ENTITY_FLG**.

Use **Program Type** to define if the algorithm's program is written using **Java**, a **Plug-In Script**, or **Java (Converted)**, meaning the program has been converted to Java.

NOTE: **Java (Converted)** program types are not applicable to all products.

Use **Program Name** to define the program to be invoked when algorithms of this type are executed:

- If the Program Type is **Java (Converted)**, enter the name of the converted program.
- If the Program Type is **Java**, enter the Java class name.
- If the Program Type is **Plug-In Script**, enter the plug-in [script](#) name. Only plug-in scripts defined for the algorithm entity may be used.

NOTE: View the source. If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#). For plug-in scripts, drill into the plug-in script to view the details.

Use the **Parameter Types** grid to define the types of parameters that algorithms of this type use. The following fields should be defined for each parameter:

- Use **Sequence** to define the relative position of the **Parameter**.
- Use **Parameter** to describe the verbiage that appears adjacent to the parameter on the Algorithm page.
- Indicate whether the parameter is **Required**. This indicator is used when parameters are defined on algorithms that reference this algorithm type.
- **Owner** indicates if the parameter for this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type with parameters. This information is display-only.

NOTE: When adding a new algorithm type that is for a Java program, the parameters are automatically generated based on the Java code. Once an algorithm type exists, any additional parameters defined in the Java code should be manually added to the algorithm type. For other program types, algorithm type parameters must be manually defined.

NOTE: When a new algorithm type parameter is added for any program type, existing algorithms for the algorithm type do not automatically get updated with the new parameter. The algorithms must be manually updated.

Where Used

An Algorithm references an Algorithm Type. Refer to [Setting Up Algorithms](#) for more information.

List of Algorithm Types

The algorithm types available to use with the product may be viewed in the algorithm type page and in the [application viewer's algorithm viewer](#). If your implementation adds algorithm types or adds algorithms to reference existing algorithm types, you may [regenerate](#) the application viewer to see your additions reflected there.

Setting Up Algorithms

If you need to introduce a new algorithm, open **Admin > System > Algorithm**. Refer to [The Big Picture Of Algorithms](#) for more information.

Description of Page

Enter an easily recognizable **Algorithm Code** and **Description** of the algorithm. **Owner** indicates if this algorithm is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new algorithm, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Reference the **Algorithm Type** associated with this algorithm.

FASTPATH: Refer to [Algorithm Type Versus Algorithm](#) for more information about how an algorithm type controls the type of parameters associated with an algorithm.

The parameters available for an algorithm are defined on the algorithm type. The system allows a set of parameter values to change over time. Use the parameter scroll to view parameter values for a given **Effective Date**. The **Owner** of the collection of parameters is displayed. The collection shows the **Parameter** description, the **Sequence** and the **Value** for each parameter.

NOTE: If the product delivers an algorithm with parameter values defined, an implementation may override the base provided parameter values by adding an additional effective dated collection of parameters.

NOTE: If an algorithm is defined and subsequently a new parameter is added to the algorithm type, existing algorithms for the algorithm type should be updated as follows: Click the "+" to add a new effective dated entry to the parameter collection. At this point the latest list of parameters for the algorithm type are visible. Configure the parameters accordingly.

Where Used

Algorithms are plugged in on control tables throughout the system. Each algorithm type's Algorithm Entity indicates the name of the control table where it is plugged in. The [algorithm](#) viewer in the application viewer may also be used to see a list of plug-in spots along with their algorithm types and algorithms.

Defining Script Options

We use the term "script" to define processing rules that your implementation sets up to control both front-end and back-end processing:

- Rules that control front-end processing are defined using [Business Process Assistant \(BPA\)](#) scripts. For example, your implementation could set up a BPA script to guide a user through your organization's payment cancellation process.
- Rules that control back-end processing are defined using [Server-based](#) scripts. For example, your implementation could set up a server-based script to control the processing that executes whenever a given type of adjustment is canceled.

The topics in this section describe how to configure your scripts.

The Big Picture Of Scripts

This section describes features and functions that are shared by both BPA scripts and server-based scripts.

Scripts Are Business Process-Oriented

To create a script, you must analyze the steps used to implement a given business process. For example, you could create a "stop auto pay" BPA script that:

- Asks the user to select the customer / taxpayer using an appropriate search page
- Asks the user to define the date on which the person would like to stop making automatic payments
- Invokes a server-based script that populates the end-date on the account's latest automatic payment instructions.

After you understand the business process, you can set up a script to mimic these steps. If the business process is straightforward (e.g., users always perform the same steps), the script configuration will be straightforward. If the business process has several logic branches, the composition of the script may be more challenging.

A Script Is Composed Of Steps

A script contains one or more steps. For example, a "stop auto pay" BPA script might have three steps:

- Ask the user to select the customer / taxpayer using an appropriate search page
- Ask the customer the date on which they'd like to stop making automatic payments (and default the current date)
- Invoke a server-based script that, in turn, updates the account's auto pay options.

Each step references a step type. The step type controls what happens when a step executes. It might be helpful to think of a script as a macro and each step as a "line of code" in the macro. Each step's step type controls the function that is executed when the step is performed.

FASTPATH: Refer to [How To Set Up Each Step Type](#) for a detailed description of all available step types and how to set them up.

Designing And Developing Scripts

Constructing a script is similar to writing a computer program. We recommend that you follow the approach outlined below when you construct scripts:

- Thoroughly understand the business process to be scripted
- Thoroughly understand how the transactions and services that your script uses work
- Design the steps for the script "on paper"
- Determine the most maintainable way to set up your scripts. Rather than creating complex, monolithic scripts, we recommend dividing scripts into smaller sections. For example:
 - For BPA scripts,
 - Determine if several scripts have similar steps. If so, set up a script that contains these common steps and invoke it from the main scripts. For example, if the main script were a BPA script, this would be invoked via a **Perform script** step.
 - Determine if a large script can be divided into logical sections. If so, set up a small script for each section and create a "master script" to invoke each. For example, if the main script were a BPA script, this would be invoked via a **Transfer control** step.
 - For server-based script, you can segregate reusable steps into "service scripts" and then use **Invoke service script** steps to execute the common logic.
- Add the script using [Script Maintenance](#)

- Thoroughly test the script

A Script May Declare Data Areas

Both BPA and server-based scripts may have one or more [data areas](#):

- If the script contains steps that exchange XML documents, you must declare a data area for each type of XML document. For example, if a BPA script has a step that invokes a service script, the BPA script must declare a data area that holds the XML document that is used to pass information to and from the service script.
- You can use a data area as a more definitive way to declare your temporary storage. For example, you can describe your script's temporary storage variables using a [stand-alone data area](#) schema and associate it with your script.

Various step types involve referencing the script's data areas as well as support the ability to compare and move data to and from field elements residing in the data areas.

An [Edit Data](#) step supports the syntax to dynamically declare data areas as part of the step itself. This technique eliminates the need to statically declare a data area. Refer to [Designing Generic Scripts](#) for an example of when this technique may be useful.

Designing Generic Scripts

Scripts may be designed to encapsulate an overall procedure common across different business objects.

For example, BPA scripts may implement a standard procedure to maintain business entities in which the first step obtains the entity's data, the second step presents its associated UI map to the user for update, and the last step updates the entity. Notice that in this case the only difference from one object to another is the data to capture. Rather than designing a dedicated BPA script with static data areas and invocation steps for each business object, you can design a single generic script that dynamically invokes a business object and its associated UI map.

This functionality is available only within an [Edit Data](#) step. With this technique, the name of the schema-based object is held in a variable or an XPath schema location and is used to both declare and invoke the object.

NOTE: Refer to [Edit Data Syntax](#) for more information on edit data commands and examples.

Securing Script Execution

The system supports the ability to secure the execution of scripts by associating the script with an Application Service. Refer to [The Big Picture of Application Security](#) for more information. Application security is optional and applies to service scripts and user-invokable BPA scripts only. If a script is not associated with an application service, all users may execute the script. Otherwise, only users that have **Execute** access to the application service may execute the script.

The Big Picture Of BPA Scripts

FASTPATH: Refer to [The Big Picture Of Scripts](#) to better understand the basic concept of scripts.

Users may require instructions in order to perform certain tasks. The business process assistant allows you to set up scripts that step a user through your business processes. For example, you might want to create scripts to help users do the following:

- Add a new person to an existing account
- Set up a customer to pay automatically
- Modify a customer who no longer wants to receive marketing information

- Modify a customer's web password
- Record a trouble order
- Merge two accounts into one account
- Fix a bill with an invalid rate
- ... (the list is only limited by your time and imagination)

Users execute these scripts via the [business process assistant](#) (BPA). Users can also define their favorite BPA scripts in their [user preferences](#). By doing this, a user can execute a script by pressing an accelerator key (Ctrl + Shift + a number).

Don't think of these scripts as merely a training tool. BPA scripts can also reduce the time it takes to perform common tasks. For example, you can write a script that reduces the "number of clicks" required to add a new person to an existing account.

CAUTION: Future upgrade issues. Although we make every effort not to remove fields or tab pages between releases, there may be times when changes made by the base-package will necessitate changes to your scripts. Please refer to the release notes for a list of any removed fields or tab pages.

CAUTION: Scripts are not a substitute for end-user training. Scripts minimize the steps required to perform common tasks. Unusual problems (e.g., a missing meter exchange) may be difficult to script as there are many different ways to resolve such a problem. However, scripts can point a user in the right direction and reduce the need to memorize obscure business processes.

The topics in this section describe background topics relevant to BPA scripts.

How To Invoke Scripts

Refer to [Initiating Scripts](#) for a description of how end-users initiate scripts.

Developing and Debugging Your BPA Scripts

You may find it helpful to categorize the step types into two groups: those that involve some type of activity in the [script area](#), and those that don't. The following step types cause activity in the script area: **Height, Display text, Prompt user, Input data, Input Map, Set focus to a field**. The rest of the step types are procedural and involve no user interaction. For debugging purposes, you can instruct the system to display text in the script area for the latter step types. Also note, for debugging purposes, you can display an entire data area (or a portion thereof) in the script area by entering `%+...+%` where ... is the name of the node whose element(s) should be displayed.

NOTE: Time saver. When you develop a new BPA script, change your [user preferences](#) to include the script as your first "favorite". This way, you can press Ctrl+Shift+1 to invoke the script (eliminating the need to select it from the [script menu](#)).

Launching A Script From A Menu

You can create menu items that launch BPA scripts rather than open a page. To do this, create a [navigation option](#) that references your script and then add a menu item that references the navigation option.

If the navigation option is referenced on a [context menu](#) and the navigation option has a "context field", a temporary storage variable will be created and populated with the unique identifier of the object in context. For example, if you add a "script" navigation option to the bill context menu and this navigation option has a context field of BILL_ID, the system will create a temporary storage variable called BILL_ID and populate it with the respective bill id when the menu item is selected.

Launching A Script When Starting The System

You can set the system to launch a script upon startup.

For example, imagine that through an interactive voice response system, a customer has keyed in their account ID and has indicated that they would like to stop an automatic payment. At the point when the IVR system determines that the customer must speak to a user, the interface can be configured to launch the application. When launched it passes the script name and account ID. It can also pass a [navigation option](#) to automatically load the appropriate page (if this information is not part of the script).

To do this, parameters are appended to the standard system URL. The following parameters may be supplied:

- `script=<scriptname>`
- `ACCT_ID=<account id>`
- `location=<navigation option>`

Parameters are added to the standard system URL by appending a question mark (?) to the end and then adding the "key=value" pair. If you require more than one parameter, use an ampersand (&) to separate each key=value pair.

For example, the following URLs are possible ways to launch the **CM-StopAutoPay** script at startup, assuming your standard URL for launching the system is `http://system-server:1234/cis.jsp`:

- `http://system-server:1234/cis.jsp?script=CM-StopAutoPay`
- `http://system-server:1234/cis.jsp?script=CM-StopAutoPay&ACCT_ID=1234512345`
- `http://system-server:1234/cis.jsp?script=CM-StopAutoPay&ACCT_ID=1234512345&location=accountMaint`

It doesn't matter in which order the parameters are provided. The system processes them in the correct order. For example, the following examples are processed by the system in the same way:

- `http://system-server:1234/cis.jsp?ACCT_ID=1234512345&script=CM-StopAutoPay&location=accountMaint`
- `http://system-server:1234/cis.jsp?ACCT_ID=1234512345&location=accountMaint&script=CM-StopAutoPay`

These parameters are kept in a common area accessible by any script for the duration of the session. To use these parameters on a script you may reference the corresponding `%PARM-<name>` [global variables](#). In this example, after the system is launched any script may have access to the above account ID parameter value by means of the `%PARM-ACCT_ID` global variable. Also note, these parameters are also loaded into temporary storage (to continue the example, there'd also be a temporary storage variable called `ACCT_ID` that holds the passed value).

Navigate to a Given Record's Maintenance Portal

If your use case is to simply navigate to the maintenance page for a given record and display that record, the script **F1-GotoPrtl** (Navigate to portal for an MO and key values) may be used. This script is only applicable to records that are governed by a business object (and define a navigation option as a BO option). It takes the incoming maintenance object code and primary key values, looks up the appropriate portal navigation option from the record's BO, populates keys into the 'page data model' and navigates to the portal.

The script expects the keys to be passed in using variable names `pkValue1` through `pkValue5`. This example navigates to the appropriate Migration Data Set portal for the given ID's business object.

- `http://system-server:1234/cis.jsp?script=F1-GotoPrtl&pkValue1=1234512345&mo=F1-MIGRDS`

Executing A Script When A To Do Entry Is Selected

The system creates [To Do entries](#) to highlight tasks that require attention (e.g., records in error). Users can complete many of these tasks without assistance. However, you can set up the system to automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You

can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

The following points provide background information to help you understand how to implement this functionality:

- Every To Do entry references a [To Do type](#) and a [message category and number](#). The To Do type defines the category of the task (e.g., bill errors). The message number defines the specific issue (e.g., a valid address can't be found.). Refer to [The Big Picture of System Messages](#) for more information about message categories and numbers.
- When a user drills down on a To Do entry, either a script launches OR the user is transferred to the transaction associated with the entry's To Do type. You control what happens by configuring the To Do type accordingly:
 - If you want to launch a script when a user drills down on an entry, you link the script to the [To Do type and message number](#). Keep in mind that you can define a different script for every message (and some To Do types have many different messages).
 - If the system doesn't find a script for an entry's To Do type and message number, it transfers the user to the To Do type's default transaction.

NOTE: How do you find the message numbers? We do not supply documentation of every To Do type's message numbers (this list would be impossible to maintain and therefore untrustworthy). The best way to determine which message numbers warrant a script is during pre-production when you're testing the system. During this period, To Do entries will be generated. For those entries that warrant a script, simply display the entry on [To Do maintenance](#). On this page, you will find the entry's message number adjacent to the description.

- These types of scripts invariably need to access data that resides on the selected To Do entry. Refer to [How To Use To Do Fields](#) for the details.

The Big Picture Of Script Eligibility Rules

You can configure [eligibility criteria](#) on the scripts to limit the scripts that a user sees in the [script search](#). For example, you could indicate a script should only appear on the script menu if the user belongs to the level 1 customer service representative group. You may also indicate that a script should only appear if the data a user is viewing has certain criteria. For example, if you are using Oracle Utilities Customer Care and Billing, you can indicate that a script should only appear if the current account's customer class is residential. By doing this, you avoid presenting the user with scripts that aren't applicable to the current data in context or the user's role.

The topics in this section describe eligibility rules.

Script Eligibility Rules Are Not Strictly Enforced

The [script search](#) gives a user a choice of seeing all scripts or only scripts that are eligible (given the current data in context and their user profile). This means that it's possible for a script that isn't eligible for the given context data / user to be executed via this search. In other words, the system does not strictly enforce a script's eligibility rules.

It might be more helpful to think of eligibility rules as "highlight conditions". These "highlight conditions" simply control whether the script appears in the [script search](#) when a user indicates they only want to see eligible scripts.

You Can Mark A Script As Always Eligible

If you don't want to configure eligibility rules, you don't have to. Simply indicate that the script is always eligible.

You Can Mark A Script As Never Eligible

If you have scripts that you do not want a user to select from the script menu, indicate that it is never eligible. An example of a script that you wouldn't want a user to select from the menu is one that is [launched when a To Do entry is selected](#).

These types of scripts most likely rely on data linked to the selected To Do entry. As a result, a user should only launch scripts of this type from the To Do entry and not from the script menu.

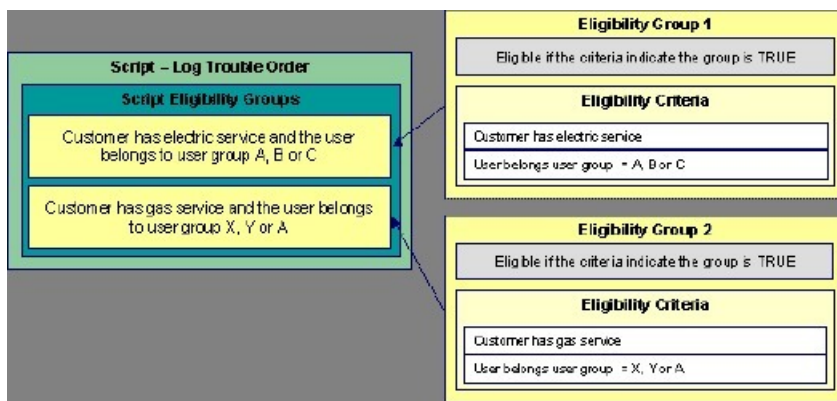
Criteria Groups versus Eligibility Criteria

Before we provide concrete examples of eligibility criteria, we need to explain two concepts: Criteria Groups and Eligibility Criteria. A script's criteria groups control whether a user is eligible to choose a script. At a high level, it works like this:

- A criteria group has one or more eligibility criteria. A group's criteria control whether the group is considered true or false.
- When you create a group, you define what should happen if the group is true or false. You have the following choices:
 - The user is eligible to choose the script
 - The user is not eligible to choose the script
 - The next group should be checked

We'll use the following example from Oracle Utilities Customer Care and Billing to help illustrate these points. Assume a script is only eligible if:

- The customer has electric service and the user belongs to user group A, B or C
- OR, the customer has gas service and the user belongs to user group X, Y or A



This script requires two eligibility groups because it has two distinct conditions:

- IF (Customer has electric service AND (User belongs to user group A, B or C))
- IF (Customer has gas service AND (User belongs to user group X, Y or A))

If either condition is true, the script is eligible.

You would need to set the following criteria groups in order to support this requirement:

Group No.	Group Description	If Group is True	If Group is False
1	Customer has electric service and the user belongs to user group A, B or C	Eligible	Check next group
2	Customer has gas service and the user belongs to user group X, Y or A	Eligible	Ineligible

The following criteria are required for each of the above groups:

Group 1: Customer has electric service and the user belongs to user group A, B or C				
Seq	Logical Criteria	If Eligibility Criteria is True	If Eligibility Criteria is False	If Insufficient Data

10	Customer has electric service	Check next condition	Group is false	Group is false
20	User belongs to user group A, B or C	Group is true	Group is false	Group is false
<hr/>				
Group 2: Customer has gas service and the user belongs to user group X, Y or A				
Seq	Logical Criteria	If Eligibility Criteria is True	If Eligibility Criteria is False	If Insufficient Data
10	Customer has gas service	Check next condition	Group is false	Group is false
20	User belongs to user group X, Y or A	Group is true	Group is false	Group is false

The next section describes how you might configure the specific logical criteria in each of the groups.

Defining Logical Criteria

When you set up an eligibility criterion, you must define two things:

- The field to be compared
- The comparison method

You have the following choices in respect of identifying the *field to be compared* :

- You can execute an algorithm to retrieve a field value from somewhere else in the system.
- Some products may support choosing a characteristic linked to an appropriate object in the system (such as an account or person).

You have the following choices in respect of identifying the *comparison method*:

- You can choose an operator (e.g., >, <, =, BETWEEN, IN, etc.) and a comparison value.
- You can execute an algorithm that performs the comparison (and returns True, False or Insufficient Data). This is also a very powerful feature, but it's not terribly intuitive. We'll present a few examples later in this section to illustrate the power of this approach.

The [Examples Of Script Eligibility Rules](#) provide examples to help you understand this design.

Examples Of Script Eligibility Rules

The topics in this section provide examples about how to set up script eligibility rules.

A Script With A Time Span Comparison

A script that is only eligible for senior citizens has the following eligibility rules:

- Customer class = Residential
- Birth date equates to that of a senior citizen

These rules require only one eligibility group on the script. It would look as follows:

Group No.	Group Description	If Group is True	If Group is False
1	Residential and Senior Citizen	Eligible	Ineligible

The following criteria will be required for this group:

Group 1: Residential, Calif, Senior					
Seq	Field to Compare	Comparison Method	If True	If False	If Insufficient Data

10	Algorithm: retrieve account's customer class	= R	Check next condition	Group is false	Group is false
30	Person characteristic: Date of Birth	Algorithm: True if senior	Group is true	Group is false	Group is false

The first criterion is easy; it calls an algorithm that retrieves a field on the current account. This value, in turn, is compared to a given value. If the comparison results in a True value, the next condition is checked. If the comparison doesn't result in a True value, the **Group is false** (and, the group indicates that if the group is false, the script isn't eligible). Refer to SECF-ACCTFLD in the product documentation for an example of an algorithm type that retrieves a field value from an account.

The last criterion contains a time span comparison. Time span comparisons are used to compare a date to something. In our example, we have to determine the age of the customer based on their birth date. If the resultant age is > 65, they are considered a senior citizen. To pull this off, you can take advantage of a comparison algorithm supplied with the base script as described below.

- **Field to Compare.** The person characteristic in which the customer's birth date is held is selected.
- **Comparison Method.** We chose a comparison algorithm that returns a value of **True** if the related field value (the customer's date of birth) is greater than 65 years (refer to [SECC-TIMESPN](#) for an example of this type of algorithm).

You'll notice that if a value of **True** is returned by the **True if senior** algorithm, the group is true (and we've set up the group to indicate a true group means the script is eligible).

NOTE: The time span algorithm can be used to compare days, weeks, months, etc. Refer to [SECC-TIMESPN](#) for more information about this algorithm.

A Script With Service Type Comparison

Imagine a script that is only eligible if the current customer has gas service and the user belongs to user groups A, B or C. This script would need the following eligibility rules:

- Customer has gas service
- User belongs to user group A, B, or C

These rules require only one eligibility group on the script. It would look as follows:

Group No.	Group Description	If Group is True	If Group is False
1	Has gas service and user is part of user group A, B or C	Eligible	Ineligible

The following criteria are required for this group:

Group 1: Has gas service and user is part of user group A, B, or C					
Seq	Field to Compare	Comparison Method	If True	If False	If Insufficient Data
10	Algorithm: check if customer has gas service	= True	Check next condition	Group is false	Group is false
20	Algorithm: check if user belongs to user group A, B or C	= True	Group is true	Group is false	Group is false

Both criteria are similar - they call an algorithm that performs a logical comparison. These algorithms are a bit counter intuitive (but understanding them provides you with another way to implement complex eligibility criteria):

The first criterion works as follows:

- **Field to Compare.** We chose a "field to compare" algorithm that checks if the current account has service agreements that belong to a given set of service types. It returns a value of **True** if the customer has an active service agreement that matches one of the service types in the algorithm. In our example, the "check if customer has gas service" algorithm

returns a value of **True** if the customer has at least one active service agreement whose SA type references the gas service type. The "check if customer has electric service" algorithm is almost identical, only the service type differs.

- Comparison Method. We simply compare the value returned by the algorithm to True and indicate the appropriate response.

The second criterion works similarly:

- Field to Compare. We chose a "field to compare" algorithm that checks if the user belongs to any user group in a set of user groups. It returns a value of **True** if the user belongs to at least one user group defined in parameters of the algorithm. Refer to [SECF-USRNGRP](#) for an example of this type of algorithm.
- Comparison Method. We simply compare the value returned by the algorithm to True and indicate the appropriate response.

NOTE: Bottom line. The "field to compare" algorithm isn't actually returning a specific field's value. Rather, it's returning a value of **True** or **False**. This value is in turn, compared by the "comparison method" and the group is set to true, false or check next accordingly.

The Big Picture Of Server-Based Scripts

FASTPATH: Refer to [The Big Picture Of Scripts](#) to better understand the basic concept of scripts.

Server-based scripts allow an implementation to configure backend business processes. The system supports two types of server-based scripts, **Plug-In** scripts and **Service** scripts.

- Plug-in scripts allow an implementation to develop routines that are executed from the system's various plug-in spots without coding. For example, an implementation could configure a plug-in script that is executed every time an adjustment of a given type is frozen.
- Service scripts allow an implementation to develop common routines that are invoked from both front-end and back-end services. For example, an implementation could create a service script that terminates an account's automatic payment preferences. This service script could be invoked from a BPA script initiated by an end-user when a customer asks to stop paying automatically, and it could also be executed from a plug-in script if a customer fails to pay their debt on time.

The topics in this section describe background topics relevant to server-based scripts.

Plug-In Scripts

NOTE: This section assumes you are familiar with the notion of plug-in spots (algorithm entities) and plug-ins. See [The Big Picture Of Algorithms](#) for more information.

Rather than write a java program for a plug-in spot, you can create a plug-in using the scripting "language". In essence, this is the only difference between a program-based plug-in and a script-based one. Obviously, this is a significant difference as it allows you to implement plug-ins without programming (and compilers).

The following topics describe basic concepts related to plug-in scripts.

A Plug-In Script's API

Like program-based plug-ins, plug-in scripts:

- Run on the application server
- Have their API (input / output interface) defined by the plug-in spot (i.e., plug-in scripts don't get to declare their own API)

- Can only be invoked by the "plug-in spot driver"

The best way to understand a plug-in script's API is to use the **View Script Schema** hyperlink to view its parameters data area schema.

```

<schema>
  <parm type="group">
    <soft type="list">
      <value/>
    </soft>
    <hard type="group">
      <action use="input"/>
      <businessObject type="group" use="input">
        <id/>
      </businessObject>
    </hard>
  </parm>
</schema>

```

Notice the two groups: soft and hard. If you are familiar with plug-in spots, you'll recognize these as the classic soft and hard parameters:

- The **soft** parameters are the values of the parameters defined on the algorithm. Notice they are not named - if you want to reference them in your plug-in script, you must do it by position (this is equivalent to what a Java programmer does for plug-ins written in Java).
- The **hard** parameters are controlled by the plug-in spot (i.e., the algorithm entity). Notice that this plug-in spot has a single input parameter called "**businessObject/id**". Also notice the **use=** attribute - this shows that this parameter is input-only (i.e., you can't change it in the plug-in script).

NOTE: XPath. You can click on an element name to see the XPath used to reference the element in your script.

Setting Up Plug-In Scripts

You can write plug-in scripts for all plug-in spots that have been Java-enabled. The following points describe how to implement a plug-in script:

- Compose your plug-in [script](#), associating it with the appropriate algorithm entity (plug-in spot).
- Create a new algorithm type for the respective algorithm entity, referencing your plug-in script as the program to carry out the algorithm type's function. Only plug-in scripts associated with the algorithm entity may be referenced on the algorithm type.
- Set up an algorithm for the respective algorithm type and plug it in where applicable. Refer to [Setting Up Algorithm Types](#) for more information.

NOTE: No plug-in scripts are shipped with the base-package.

Service Scripts

BPA scripts run on the client's browser and guide the end-users through business processes. Service scripts run on the application server and perform server-based processing for [BPA scripts](#), [zones](#) and more. You may want to think of a service script as a common routine that is set up via the scripting (rather than programming).

The following topics describe basic concepts related to service scripts.

A Service Script's API

As with any common routine, a service script must declare its input / output parameters (i.e., its API). A service script's API is defined on its [schema](#).

NOTE: Refer to [Schema Nodes and Attributes](#) for a complete list of the XML nodes and attributes available to you when you construct a schema.

Invoking Service Scripts

Any type of script may [invoke a service script](#):

- A BPA script may invoke a service script to perform server-based processing.
- Plug-in scripts may invoke a service script (like a "common routine").
- A service script may call another service script (like a "common routine").

Map zones may be configured to invoke service scripts to obtain the data to be displayed. Refer to [Map Zones](#) for more information.

[Inbound web services](#) support interaction with service scripts allowing the outside world to interact directly with a service script.

You can also invoke a service script from a Java class.

Debugging Server-Based Scripts

The server can create log entries to help you debug your server scripts. These logs are only created if you do the following:

- Start the system in **?debug=true** mode
- Turn on the **Global debug** checkbox in the upper corner of the browser. When you click this check box, a pop-up appears asking you what you want to trace - make sure to check box that causes script steps to be traced.

The logs contain a great deal of information including the contents of the referenced data area for **Move data**, **Invoke business object**, **Invoke business service** and **Invoke service script** steps.

You can view the contents of the logs by pressing the **Show User Log** button appears at the top of the browser.

Please note that all log entries for your user ID are shown (so don't share user IDs).

Maintaining Scripts

The script maintenance transaction is used to maintain your scripts. The topics in this section describe how to use this transaction.

FASTPATH: Refer to [The Big Picture Of Scripts](#) for more information about scripts.

Script - Main

Use this page to define basic information about a script. Open this page using **Admin > System > Script**.

NOTE: Script Tips. A context sensitive "Script Tips" zone is associated with this page. The zone provides links to [Edit Data Syntax](#) and [Advanced Schema Topics](#) so that users can quickly access those online help topics to aid in constructing scripts.

Description of Page

Enter a unique **Script** code and **Description** for the script. Use the **Detailed Description** to describe the purpose of this script in detail. **Owner** indicates if the script is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new script, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

Script Type indicates if this is a **BPA Script**, **Plug-In Script** or **Service Script**. Refer to [The Big Picture Of BPA Scripts](#) and [The Big Picture Of Server Based Scripts](#) for more information.

Accessibility Option appears only for BPA scripts. Set this value to **Accessible from Script Menu** for any script that may be launched as a stand-alone script. Scripts with this configuration may be linked to a navigation option so that they may be invoked from a menu and may be configured by a user as a favorite script. Set this value to **Not Accessible from Script Menu** for any script that cannot be launched on its own. For example, any script that is invoked as a sub-script from another script should have this setting. In addition, any script that is designed to be launched from within a specific portal where certain data is provided to the script should include this setting.

Enter an **Application Service** if the execution of the script should be secured. The application service should include **Execute** as one of its access modes. Refer to [Securing Script Execution](#) for more information.

Algorithm Entity appears only for [plug-in scripts](#). Use this field to define the [algorithm entity](#) into which this script can be plugged in.

Business Object appears only for business object related plug-in scripts. Enter the [Business Object](#) whose elements are to be referenced by the plug-in script.

Script Engine Version defines the version of the XML Path Language (XPath) to be used for the script. Script engine versions 2 and 3 use the XPath 2 engine supplied by the XQuery team. This is the same engine used inside the Oracle database. The current script engine version 3 is a modified version that offers performance improvements without impacting existing version 2 scripts.

Notes regarding Script engine version 1: The XPath library used is Jaxen; for BPA scripts, use IE's built-in MSXML parser; Xpath 1 (and even JavaScript) uses floating point arithmetic, which means that adding a collection of numbers with two decimal places might end up with a value of 10779.079999999998 instead of 10779.08.

Click the **View Script Schema** to view the [script's data areas](#) on the [schema viewer](#) window.

Click the **View XSD** hyperlink to view a service script's schema definition in XSD format.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window. The presented script syntax is valid within [edit data](#) steps.

The [tree](#) summarizes the script's steps. You can use the hyperlink to transfer you to the **Step** tab with the corresponding step displayed.

Script - Step

Use this page to add or update a script's steps. Open this page using **Admin > System > Script** and then navigate to the **Step** tab.

NOTE: Time saver. You can navigate to a step by clicking on the respective node in the tree on the Main tab.

Description of Page

The **Steps** accordion contains an entry for every step linked to the script. When a script is initially displayed, its steps are collapsed. To see a step's details, simply click on the step's summary bar. You can re-click the bar to collapse the step's details. Please see [accordions](#) for the details of other features you can use to save time.

Select the **Step Type** that corresponds with the step. Refer to [How To Set Up Each Step Type](#) for an overview of the step types.

CAUTION: The Step Type affects what you can enter on other parts of this page. The remainder of this section is devoted to those fields that can be entered regardless of Step Type. The subtopics that follow describe those fields whose entry is contingent on the Step Type.

Step Sequence defines the relative position of this step in respect of the other steps. The position is important because it defines the order in which the step is executed. You should only change a Step Sequence if you need to reposition this step. But take care; if you change the Step Sequence and the step is referenced on other steps, you'll have to change all of the referencing steps.

NOTE: Leave gaps in the sequence numbers. Make sure you leave space between sequence numbers so that you can add new steps between existing ones in the future. If you run out of space, you can use the **Renumber** button to renumber all of the steps. This will renumber the script's steps by 10 and change all related references accordingly.

Display Step is only enabled on BPA scripts for step types that typically don't cause information to be displayed in the [script area](#) (i.e., step types like **Conditional Branch**, **Go to a step**, **Height**, etc). If you turn on this switch, information about the step is displayed in the script area to help you debug the script.

CAUTION: Remember to turn this switch off when you're ready to let users use this script.

NOTE: If **Display Step** is turned on and the step has **Text**, this information will be displayed in the script area. If **Display Step** is turned on and the step does NOT have **Text**, a system-generated messages describing what the step does is displayed in the script area.

Display Icon controls the [icon](#) that prefixes the **Text** that's displayed in the script area. Using an icon on a step is optional. This field is only applicable to BPA scripts.

Text is the information that displays in the [script area](#) when the step executes. You need only define text for steps that cause something to display in the script area.

FASTPATH: Refer to [How To Substitute Variables In Text](#) for a discussion about how to substitute variables in a text string.

FASTPATH: Refer to [How To Use HTML Tags And Spans In Text](#) for a discussion about how to format (with colors and fonts) the text that appears in the script area.

The other fields on this page are dependent on the **Step Type**. The topics that follow briefly describe each step type's fields and provide additional information about steps.

Click on the **View Script Schema** hyperlink to view the script's data areas. Doing this opens the [schema viewer](#) window.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window. The presented script syntax is valid within [edit data](#) steps.

How To Set Up Each Step Type

The contents of this section describe how to set up each type of step.

Common Step Types To All Script Types

The contents of this section describe common step types applicable to all script types.

How To Set Up Conditional Branch Steps

Conditional branch steps allow you to conditionally jump to a different step based on logical criteria. For example, you could jump to a different step in a script if the customer is residential as opposed to commercial. In addition, several fields are required for **Conditional Branch** steps:

Compare Field Type and **Compare Field Name** define the first operand in the comparison. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Current To Do Information.** Use this field type when the field being compared resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.
- **Data Area.** Use this field type when the field being compared is one that you put into one of the scripts data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the field being compared resides on one of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Predefined Value.** Use this field type when the field being compared is a [global variable](#).
- **Temporary Storage.** Use this field type when the field being compared is one that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field.** Use this field type when the field being compared resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

Condition defines the comparison criteria:

- Use **>**, **<**, **=**, **>=**, **<=**, **<>** (not equal) to compare the field using standard logical operators. Enter the comparison value using the following fields.
- Use **IN** to compare the first field to a list of values. Each value is separated by a comma. For example, if a field value must equal **1**, **3** or **9**, you would enter a comparison value of **1,3,9**.
- Use **BETWEEN** to compare the field to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.

Comparison Field Type, **Comparison Field Name** and **Comparison Value** define what you're comparing the first operand to. The following points describe each field type:

- **Current To Do Information.** Use this field type when the comparison value resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.
- **Data Area.** Use this field type when the comparison value resides in one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the comparison value resides on one of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Predefined Value.** Use this field type when the field being compared is a constant value defined in the script. When this field type is used, use **Comparison Value** to define the constant value. Refer to [How To Use Constants In Scripts](#) for instructions on how to use constants.
- **Temporary Storage.** Use this field type when the comparison value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.

- **User Interface Field.** Use this field type when the comparison value resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

NOTE: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

The above fields allow you to perform a comparison that results in a value of **TRUE** or **FALSE**. The remaining fields control the step to which control is passed given the value:

- **If TRUE, Go to** defines the step that is executed if the comparison results in a **TRUE** value.
- **If FALSE, Go to** defines the step that is executed if the comparison results in a **FALSE** value.

NOTE: Numeric Comparison. Comparison of two values may be numeric or textual (left-to-right). Numeric comparison takes place only when values on both side of the comparison are recognized as numeric by the system. Otherwise, textual comparison is used. Fields for **Current To Do Information**, **Data Area**, **Page Data Model**, and **User Interface Field** types are explicitly associated with a data type and therefore can be recognized as numeric or not. This is not the case for fields residing in **Temporary Storage** or those set as **Predefined Values**. A **Temporary Storage** field is considered numeric if it either holds a numeric value moved to it from an explicitly defined numeric value (see above) or it is a resultant field of mathematical operation. A **Predefined Value** field is considered numeric if the other field it is compared to is numeric. For example, if a numeric field is compared to a **Predefined Value** the latter is considered numeric as well resulting in numeric value comparison. However, if the two fields are defined as **Predefined Values** the system assumes their values are text strings and therefore applies textual comparison.

How To Set Up Edit Data Steps

Edit data steps provide a free format region where you can specify commands to control your script processing.

In general, the syntax available within edit data mimics the commands available within the explicit step types. However, there are a few commands that are available only within edit data. For example, the two structured commands: **For**, and **If**.

For server-based scripts, you may find it useful to create a few explicit step types and then use the **View Script as Text** hyperlink on the [Script - Step](#) page to better understand the edit data syntax.

NOTE: Not all BPA step types are supported using the edit data syntax. Refer to the Edit Data Syntax topic below for more information on edit data commands and examples.

Additional field required for **Edit data** steps:

Enter your scripting commands in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for defining the edit data step.

Edit Data Syntax

The topics in this section provide detail of the syntax supported in the edit data step type.

Contents

- [Comments](#)
- [Temporary Variables](#)
- [Context Variables](#)
- [Move Statement](#)
- [Go To Statement](#)
- [Conditional Branch Statement](#)
- [If Statement](#)
- [For Statement](#)
- [List Processing](#)
- [Functions for Processing a List](#)
- [Declare and Invoke Schema Based Objects](#)
- [System and Global Variables](#)
- [Perform Script and Transfer](#)
- [Navigate Statement](#)
- [Invoke Map Statement](#)
- [Declare BO with BO Group](#)
- [Generate Edit Map Statements](#)
- [Terminate Statement](#)
- [Debugging a BPA Script](#)

Comments

You can place comments into your script using the double slash notation `//` in the first two characters of the edit data step. Example:

```
//  
// quit with error  
//  
if ("not(customer/securityEnabled)")  
terminate with error (8000, 1001 %1="customer/id" %1='not allowed');  
end-if;
```

Temporary Variables

Temporary variables can be declared within all types of scripts. They should be referenced by a leading single dollar sign (`$`). However, temporary variables behave differently in the various script types:

- In BPA Scripts temporary variables remain persistent from one BPA script to another (refer to the [Perform Script and Transfer Control statements](#)), which means that you can use temporary variables to communicate between BPA scripts.
- Temporary variables cannot be passed from a BPA script to a service or plug-in script. Only data area elements can be passed between these types of scripts.
- Within service and plug-in scripts, temporary variables remain persistent only for the life of the particular script that declared the variable. This means that temporary variables cannot be passed between plug-in scripts and service scripts, only [global variables](#), [context variables](#) and data area elements can be passed between these types of scripts.

Declaring / Initializing / Defaulting Temporary Variables

When using a temporary variable, it should be declared or initialized with an appropriate value before using it. A typical method for declaring a variable is to simply move data to it in a move statement, for example.

```
move "0" to $index;
```

FASTPATH: Refer to [Move to a Temporary Variable](#) for more information on implicit declaration of a temporary variable within a move statement.

For BPA scripts, as mentioned above, temporary variables may be passed from one BPA script to another. As such, it is common to reference a temporary variable in a BPA that should have been initialized by a previous BPA. However, if there is any reason that a temporary variable did not get initialized by a previous BPA, a reference to it will cause an error. It is

good practice, therefore, to use the **default** statement that will initialize temporary variables that are not already created / initialized.

- The following statement will initialize the temporary variable \$InputAction – but only if the temporary variable has not yet been initialized:

```
default $InputAction;
```

- The following statement will set the value of the temporary variable \$InputAction to 'read' – but only if the variable has not yet been initialized:

```
default $InputAction as 'read';
```

NOTE: Scripts should take care not to define variables using a reserved keyword. The following table lists the reserved keywords.

Keyword

add

as

asError

bpa

branch

data

declareBO

declareBS

declareDA

declareMap

declareSS

default

delete

edit

element

else

end-edit

end-for

end-if

error

escape

evaluate

fastAdd

fastUpdate

for

goto

if

in

invokeBO

Keyword

invokeBS

invokeMap

invokeSS

label

map

move

navigate

navigateAndReloadDashboard

null

page

performScript

popup

read

readWithoutVersion

replace

suppress

target

terminate

to

transferControl

update

using

warn

with

Context Variables

Context variables are only available within service scripts. The context variable will be available for the duration of the service script and all that it invokes. Therefore, you can use a context variable within a service script to communicate information to a lower level service script or schema. They should be referenced by leading double dollar signs ('\$\$').

NOTE: Because context variables are available for lower level scripts, they may sometimes be referred to as global variables or global context variables. But they should not be confused with [global variables](#).

Declaring / Initializing / Defaulting Context Variables

When using a context variable, it should be declared or initialized with an appropriate value before using it. A typical method for declaring a variable is to simply move data to it in a move statement, for example.

```
move 'context variable' to $$contextVariable;
```

FASTPATH: Refer to [Move using a Context Variable](#) for more information.

Move Statement

The **move** statement copies a source value to a target. The following table highlights various options supported in the move statement.

Statement	Example Description	Example Syntax
Move to Element move "xpath" to "xpath";	Move statement with simple XPath reference.	<pre>move "acct/totalBalance" to "parm/formattedValue";</pre>
NOTE: An XPath expression is surrounded by double quotes.	Move statement with XPath concatenate function.	<pre>move "concat(person/firstName, ', ', person/lastName)" to "parm/fullName";</pre>
	Move statement with XPath substring-before function.	<pre>move "substring-before(parm/fullName,', ')" to "person/firstName";</pre>
	Move statement with XPath substring-after function.	<pre>move "substring-after(parm/fullName,', ')" to "person/lastName";</pre>
	Move statement with XPath substring function.	<pre>move "substring(parm/date,1,4)" to "parm/year";</pre>
Move to Element move 'literal' to "xpath";	Move statement using a literal string.	<pre>move 'okay for mailing' to "account/preferences[type='mail']/text";</pre>
NOTE: A literal value is surrounded by single quotes.		
Move to Element move 'Boolean' to "xpath";	Move statement with Boolean as literal string.	<pre>if ("account/balance > 0") move 'true' to "account/hasDebitBalance"; end-if;</pre>
	Moving an expression, which results in a Boolean. Note that the filter in the example below is located on a group node.	<pre><schema> <account> <hasDebitBalance type="boolean"/> > <balance/> </account> </schema></pre>
		<pre>move "boolean(account[balance>0])" to "account/hasDebitBalance";</pre>
Move to Group move "xpath" to "xpath";	Move a set of elements from one group with another – where the element names are valid for the target schema.	<pre>move "account/custInfo" to "person";</pre> <p>This statement is equivalent to the following</p> <pre>move "account/custInfo/*" to "person/*";</pre>
Move using a Temporary Variable	When moving to a temporary local variable, it is not surrounded by double quotes. move "xpath" to \$variable;	<pre>move "count(Person/names/personName) + count(Person/ids/personId)" to \$PersonChildCount;</pre>
	When moving from a temporary variable, the variable is surrounded by double quotes. move "\$variable" to "xpath";	<pre>move "\$AccountBalance" to "parm/formattedValue";</pre>
Move using a Context Variable move "xpath" to \$\$variable; move \$\$variable to "xpath";	Context variables, source or target, are referenced without any double quotes.	<pre>move 'context value' to \$contextVariable; // // here, we move from a context variable.</pre>

Statement	Example Description	Example Syntax
		<pre>move \$ \$contextVariable to "MarketMessage/ sender";</pre>
Move using a Dynamic Location move "xpath" 'literal' to evaluate("xpath" \$variable); move evaluate("xpath" \$variable) to "xpath" \$variable;	The evaluate statement allows your move source or target location to be dynamically derived from a variable or schema element location.	<pre>move 'literal' to evaluate("schemaLocation") // move "schemaLocation" to evaluate(\$Variable) move evaluate("schemaLocation") to \$Variable // move evaluate(\$Variable) to "schemaLocation"</pre>
Move escape move escape("xpath" \$variable) to "xpath" \$variable;	Move escape is only available for service scripts and plug-in scripts. The escape statement scans your source text value for HTML content and escapes it, i.e. replaces any HTML-like characters with special characters that are escaped from HTML rendering. By doing so the text would be displayed as plain text when displayed as part of an HTML element. NOTE: You should only use this function if the text is to be displayed as part of an HTML element and is suspected to contain HTML-like characters or even malicious HTML that should not be rendered as HTML. If incorrectly displayed using a non HTML element, the special escape characters, if any would be visible as part of your text. Refer to the UI Map Attributes and Functions for more information on how to define an element to display HTML content.	<pre>move escape("schemaLocation") to \$Variable; // move escape(\$Variable) to "schemaLocation";</pre>
Move null move null to "xpath";	You can remove information from the XML instance document through the special syntax of move 'null'. Note that you can specify either a node name in the XPath expression or a group name. If you specify a group then the group and all child elements will be eliminated from processing.	Remove a node and all of its child nodes: <pre>if ("boolean(customer/ securityEnabled)") goto updateInfo; else move null to "customer"; end-if;</pre> Remove all child nodes of a group node with the suffix '/*'. <pre>if ("boolean(customer/ securityEnabled)") move null to "customer/*"; end-if;</pre>

Go To Statement

The edit data step supports functionality analogous to the [Go To](#) step type. The syntax is **goto label;** where the label represents another location within the edit data text field (identified by this label) or represents another step in the script.

The following is an example of going to another location in the same step identified by the label **addSpouse**.

```
if ("string(parm/spouse/name) != $BLANK")
goto addSpouse;
end-if;
addSpouse: invokeBO 'Person' using "parm/spouse" for add;
```

The following is an example of going to a different step within the same script. The step sequence is the reference used as the label.

```
if ("string(parm/spouse/name) != $BLANK")
  goto 110;
end-if;
.
.
.
110: invokeBO 'Person' using "parm/spouse" for add;
```

Conditional Branch Statement

The edit data step supports functionality analogous to the [Conditional Branch](#) step type. The syntax is **branch (“xpath”) goto label else label;** where:

- The XPath condition in the **branch** statement must evaluate to a Boolean value of True or False.
- The targets for the **goto** and **else** statements are labels that represent another location within the edit data text field (identified by this label) or represent another step in the script.

The following example uses labels for **addSpouse** and **addAccount**

```
branch ("string(parm/spouse/name) != $BLANK") goto addSpouse else addAccount;
```

If Statement

The **if** statement is similar to the conditional branch statement. Either can be used to structure the logic of your script. This statement may optionally include an **else** statement but it should always end with an **end-if** statement.

NOTE: This is an example of a statement that is not represented as a separate step type. It is only available within the edit data text.

The syntax is **if (“xpath”) else end-if;**. The XPath condition must evaluate to a Boolean value of True or False. The following are some examples.

Example where the XPath contains a simple logical condition.

```
if ("string(parm/spouse/name) != $BLANK")
  //
  // Create spouse since spouse name present
  goto addSpouse;
else
  //
  // Create account without spouse
  goto addAccount;
end-if;
```

Example where the XPath contains a complex condition.

```
if ("string(parm/spouse/name) != $BLANK and string(parm/hasSpouse) = true or boolean(parm/requireSpouse)")
  //
  // Create spouse since spouse name present
  goto addSpouse;
end-if;
```

Example of a stacked set of statements used to evaluate multiple possible values of a field.

```
if ("parm/rowCount = 0")
  //
  // no rows found
  goto quit;
end-if;
if ("parm/rowCount = 1")
  //
  // one row found
  goto process;
end-if;
```



```

if ("parm/rowCount > 1")
  //
  // more than one row found
  goto quit;
end-if;
quit: terminate;

```

The following XPath shows Boolean based on the existence of the node. In this example, if the node exists in the XML instance document being processed, the statement will evaluate to True. If no element is found, the statement evaluates to false.

NOTE: When treating XPath nodes as Boolean variables be aware that an empty node evaluates to True. Only a missing node return False.

```

if ("boolean(parm/spouse/name)")
  goto addSpouse;
else
  //
  // Create account without spouse
  goto addAccount;
end-if;

if ("not(parm/spouse/name)")
  //
  // Create account without spouse
  goto addAccount;
else
  goto addSpouse;
end-if;

```

For Statement

The **for** statement creates a list of nodes or values depending on your XPath expression. If you specify a list node then every child node of the list, along with its contents, will be available within the loop. If you specify a child node directly, then a list of values only will be available within the loop.

NOTE: For more information on creating new entries in a list, please refer to the [creating a new list instance](#) example.

NOTE: This is an example of a statement that is not represented as a separate step type. It is only available within the edit data text.

The syntax is **for (\$variable in "xpathList") end-for;**. The XPath condition must evaluate to a Boolean value of True or False.

The following examples are based on this sample schema:

```

<schema>
  <SAList type="list">
    <id/>
    <balance/>
  </SAList>
  <SAContributor type="list">
    <id/>
    </SAContributor>
</schema>

```

Example that specifies the list node in the XPath expression where all child nodes are available for processing.

```

move "0" to $AccountBalance;
move "0" to $index;
for ($SAList in "SAList")
  move "$SAList/balance + $AccountBalance" to $AccountBalance;
  //
  // keep track of each SA contributing to the balance in the SA Contributor list
  move "1 + $index" to $index;
  move "$SAList/id" to "SAContributor[$index]/id";

```

```
end-for;
```

Example that specifies a child node within the list node in the XPath expression. Only values of that node are available for processing.

```
move "0" to $AccountBalance;
for ($SABalance in "SAList/balance")
  move "$SABalance + $AccountBalance" to $AccountBalance;
end-for;
```

Example that shows that a filter can be used to limit the rows selected by the **for** loop.

```
move "0" to $AccountDebitBalance;
for ($SAList in "SAList[Balance>0]")
  move "$SAList/balance + $AccountDebitBalance" to $AccountDebitBalance;
end-for;
```

Example that shows the use of a filter when specifying child nodes.

```
move "0" to $AccountCreditBalance;
for ($SABalance in "SAList[Balance<0]/balance")
  move "$SABalance + $AccountCreditBalance" to $AccountCreditBalance;
end-for;
```

List Processing

This section provides details about processing lists. The examples in this section reference the following schema:

```
<schema>
  <parm type="group">
    <name/>
  </parm>
  <Person type="group">
    <names type="list">
      <type/>
      <name/>
    </names>
  </Person>
</schema>
```

Referencing a List Element. You can move a value to a particular list instance by referencing an identifying node in the list within a filter. The syntax is **move "xpath" to "xpathList[filter]/element"**; Example:

```
move "parm/name" to "Person/names[type='main']/name";
```

Creating a New List Instance. A special notation can be used within a move target statement to indicate a new list instance should be created. The "+" indicates to the script processor that a new instance of a list should be initiated for the target element. The syntax is **move "xpath" to "+xpathList"**; Example:

```
move "parm/name" to "Person/+names/name";
```

Deleting a List Instance. An XML list entry can be deleted from the database by moving an action attribute of 'delete' to the element name. To cause a database delete of a list entry requires an attribute of action="delete" in the target node and a subsequent update BO interaction. The syntax is **move 'delete' to "xpathList@action"**; Example:

```
if ("parm/action = 'd'")
  move "0" to $index;
  for ($CCList in "CCList")
    move "1 + $index" to $index;
    if ("CCList/id = parm/id")
      move 'delete' to "CCList[$index]@action";
      goto update;
    end-if;
  end-for;
end-if;
```

The following shows the resulting XML.

```
<root>
  <CCList>
    <id>9876538976</id>
    <balance>309.98</balance>
```

```

</CCList>
<CCList action="delete">
  <id>4321125899</id>
  <balance>87.45</balance>
</CCList>
</root>

```

NOTE: Deleting a list instance through use of the action attribute is risky if iterative BO interactions are required. The XML document that contains the list instance to be deleted will not be altered after a successful BO interaction, which means the document will still contain the list instance even though it no longer exists. To solve this problem, it is essential to re-read the BO after any BO update where the action attribute of 'delete' has been used.

NOTE: An alternative to the delete attribute described here, is to use the BO action of [replace](#). Manipulating a list to use the replace action avoids the problem described above concerning stale information in request documents post BO update.

Functions for Processing a List

XPath provides several functions that are useful to process elements of a list including **count**, **sum** and **last**.

The following examples are based on this sample XML document:

```

<xml>
  <ft>
    <type>bill</type>
    <date>20100101</date>
    <amt>30.30</amt>
    <cat>tax</cat>
  </ft>
  <ft>
    <type>adj</type>
    <date>20100301</date>
    <amt>20.20</amt>
    <cat>int</cat>
  </ft>
  <ft>
    <type>bill</type>
    <date>20100201</date>
    <amt>10.10</amt>
    <cat>tax</cat>
  </ft>
</xml>

```

The following is an example of a sum. The syntax is **move "sum(xpathList/element)" to \$variable**; The example sums the total balance.

```
move "sum(ft/amt)" to $TotalBalance;
```

The following is an example of a sum using a filter to get a subtotal. The example sums the balance of the entries that have the 'tax' category.

```
move "sum(ft[cat='tax']/amt)" to $TaxBalance;
```

The following is an example of a count. The syntax is **move "count(xpathList)" to \$variable**; The example finds the count of the number of FT entries in the list.

```
move "count(ft)" to $TranCount;
```

The following is an example of 'last', which is used to locate the last entry. The syntax is **move "last(xpathList)" to \$variable**; The example finds the last amount in the FT list.

```
move "ft[last()]/amt" to $LastAmount;
```

Declare and Invoke Schema Based Objects

You can invoke a business object, business service or service script within the edit data step. To support the dynamic invoke, a dynamic data area name can be declared.

The schema being declared may be a business object (BO) schema, a business service (BS) schema, a service script (SS) schema, data area (DA) schema or a UI map schema. The declare statement will differ based on the type of schema, but the syntax is analogous.

- **declareBO** 'BO Name' | \$variable | "xpath" as 'DynamicDataArea';
- **declareBS** 'BS Name' | \$variable | "xpath" as 'DynamicDataArea';
- **declareSS** 'SS Name' | \$variable | "xpath" as 'DynamicDataArea';
- **declareDA** 'DA Name' | \$variable | "xpath" as 'DynamicDataArea';
- **declareMap** 'Map Name' | \$variable | "xpath" as 'DynamicDataArea';

When invoking a BO, BS or SS, the name of the object can be specified as a literal or it can be a value contained within an element or a variable. For every Invoke, you must supply an XPath reference to a group name.

- When invoking a business object, an action must be supplied. The syntax is **invokeBO** 'BO Name' | \$variable | "xpath" using "xpath" for action; The valid actions are as follows:
 - **read**. This action reads the current view of the BO data.
 - **add**. This action will add the object and read and return the resulting view of the BO.
 - **fastAdd**. This action will add the object but does not perform a subsequent 'read' to return the resulting view of the BO.
 - **update**. This action will update the object and read and return the resulting view of the BO. This action executes a 'merge' of the information specified in the invoke statement's request XML document with existing BO data. Using this action allows the script to only indicate the elements that are changing.
 - **fastUpdate**. This action will update the object but does not perform a subsequent 'read' to return the resulting view of the BO.
 - **delete**. This action deletes the object.
 - **replace**. This action is an alternate to the update action. The replace action completely replaces existing BO data with the information in the request document. Typically, the replace action is used when a BO contains a list because it is easier to simply replace all instances of a list rather than attempt a list merge, which requires special logic to [delete a list instance](#) explicitly.

NOTE: The replace action must be used when using the UI map functionality to [Upload a CSV File](#).

Examples:

```
invokeBO 'BusinessObject' using "dataArea" for fastAdd;
invokeBO $variableBO using "dataArea" for fastUpdate;
invokeBO "daName/boElement" using "dataArea" for replace;
```

- The syntax of the invoke statements for both a business service and service script are similar. The BS / SS is specified along with the XPath reference to the group name:
 - **invokeBS** 'BS Name' | \$variable | "xpath" using "xpath";
 - **invokeSS** 'SS Name' | \$variable | "xpath" using "xpath";

The examples use the **invokeBS** statement but the statements are similar for the **invokeSS** statement.

```
invokeBS 'BusinessService' using "dataArea";
invokeBS $variableBS using "dataArea";
invokeBS "daName/bsElement" using "dataArea";
```

Note that for BPA scripting, the **invoke** statements may also indicate how to handle warnings. The statement should include **with warn asError | popup | suppress**;

- **with warn asError** indicates that a warning returned from the invoke statement will be treated as an error displayed in the UI map. The text **asError** is optional. If the text **with warn** is defined with no further definition, the suffix **asError** is applied.
- **with warn popup** indicates that a warning returned from the invoke statement will be presented in the standard framework popup.
- **with warn suppress** indicates that a warning returned from the invoke statement will be suppressed. This is the default if no **with warn** is added to the **invoke** statement.

NOTE: For service scripts, all objects invoked from the service script will inherit their warning level. Therefore, if the service script is invoked **with warn**, all nested invoke statements will also be invoked **with warn**.

Examples:

```
invokeBO 'BusinessObject' using "dataArea" for add with warn asError;
invokeBS "daName/bsElement" using "dataArea" with warn popup;
invokeSS 'ServiceScript' using "dataArea" with warn;
```

For BPA scripts, there should also be logic following the invocation to handle errors and warnings (if desired). The system variables **\$ERROR** and **\$WARNING** are provided to interpret the results. Also note that the product provides a BPA Script **F1-HandleErr** that may be used to display the error. The following is an example of

```
invokeBO "F1-DetermineBo/output/bo" using "boSchema" for update with warn popup;
if (" $WARNING")
  terminate;
end-if;
if (" $ERROR")
  transferControl 'F1-HandleErr';
end-if;
```

System and Global Variables

The following tables highlight system and global variables available for script writing.

System Variables — All Script Types

The following system variables are available for all script types (service scripts, plug-in scripts, and BPA scripts).

Variable	Description	Example
\$BLANK	Represents an empty node.	<pre>if ("string(parm/spouse/name) != \$BLANK") goto addSpouse; end-if;</pre>
\$CURRENT-DATE	Represents the current date. For BPA scripts, this is the browser date. For server scripts this is the server date (and is affected by the system date override logic).	<pre>move "\$CURRENT-DATE" to \$tempDate;</pre>
\$CURRENT-STD-DTTM	Represents the current date-time expressed in standard time (meaning without any adjustments for summer time / daylight savings time).	<pre>move "\$CURRENT-STD-DTTM" to \$tempDateTime;</pre>
\$DEVICE-OS	Represents the user's device operating system.	<pre>move "\$DEVICE-OS" to \$tempDeviceOs;</pre>
\$DEVICE-BROWSER	Represents the user's device browser.	<pre>move "\$DEVICE-BROWSER" to \$tempDeviceBrowser;</pre>
\$DEVICE-DISPLAY-TYPE	Represents the user's device screen display type whether it is Desktop size or Medium or Small	<pre>move "\$DEVICE-DISPLAY-TYPE" to \$tempDeviceDisplayType;</pre>

Variable	Description	Example
	size. Returned values may be like oraDesktop, oraTablet and oraPhone.	
\$DEVICE-INFO	Provides the combination of all three device properties (DEVICE-OS, DEVICE-BROWSER and DEVICE-DISPLAY-TYPE) and each property value is separated by semi-colon.	<code>move "\$DEVICE-INFO" to \$tempDeviceInfo;</code>

System Variables — BPA Scripts Only

The following system variables are only available / applicable for BPA script types.

Variable	Description	Example
\$DOUBLE_QUOTE	Represents a double quote.	<code>move "\$DOUBLE_QUOTE" to \$tempField;</code>
\$SINGLE_QUOTE	Represents an apostrophe.	<code>move "\$SINGLE_QUOTE" to \$tempField;</code>
\$SPACE	Contains a single space value.	<code>move "\$SPACE" to \$tempField;</code>
\$SYSTEM-DATE	Represents the server date. Note that this date is affected by the system date override logic)	<code>move "\$SYSTEM-DATE" to \$tempDate;</code>

System Variables — Server Scripts Only

The following system variables are only available / applicable for service script and plug-in script types.

Variable	Description	Example
\$ADDITIONAL-IP-INFO	An HTTP request includes an "additional IP address" header field. This may be populated by an implementation if there is some information available on the proxy server or load balancer, such as the originating IP address.	<code>move "\$ADDITIONAL-IP-INFO" to "parm/request/headerIpAddress";</code>
\$CURRENT-DTTM	Represents the current date-time.	<code>move "\$CURRENT-DTTM" to \$tempDateTime;</code>
\$F1-INSTALLATION-TIMEZONE	Represents the time zone code defined on the installation options .	<code>move "\$F1-INSTALLATION-TIMEZONE" to \$timeZone;</code>
\$LANGUAGE	Represents the language code the script is using. Typically this is the user's default language.	<code>move "\$LANGUAGE" to \$tempLanguage;</code>
\$PROCESS-DATE	Represents the process date. The process date differs from the current date because the process date will remain consistent throughout the duration of the process being executed. For example, if a service script stores several business objects – the process date is initialized at the start of the service script execution and each business object will have the same process date defaulted. The current date, especially the current date time, will reflect the actual time of processing.	<code>move "\$PROCESS-DATE" to \$tempDate;</code>
\$PROCESS-DTTM	Represents the process date-time. Note that the process date and time is initialized at the start of	<code>move "\$PROCESS-DTTM" to \$tempDateTime;</code>

Variable	Description	Example
	a particular process and will not reflect the exact date and time of an update.	
\$REQUESTING-IP-ADDRESS	Represents the IP address from the HTTP request. Note that if the request is routed through a proxy server or load balancer, this IP address is be the IP address of the proxy or load balancer, not the IP address of the end user. Refer to the \$ADDITIONAL-IP-INFO variable for information.	<pre>move "\$REQUESTING-IP-ADDRESS" to "parm/request/systemIpAddress";</pre>
\$USER	Represents the user ID of the user executing the script.	<pre>move "\$USER" to \$tempUser;</pre>

Global Variables

BPA scripts and service scripts have access to the values defined in [Global Context](#).

When a BPA script is launched from the user interface, these variables will be automatically initialized. They may be referenced with a single dollar sign in front of the field name. For example if PER_ID is a supported global variable, then \$PER_ID can be referenced within the BPA script:

```
move "$PER_ID" to "schema/customerId";
```

For service scripts, global variables may only be referenced if the service script has been invoked directly from a BPA script or a zone on a portal. When a service script is invoked from a BPA script or portal zone, it will have access to the suite of global context variables populated in the UI session. For service scripting, the global fields must be prefixed by two dollar signs (instead of one like in BPA scripting). For example if PER_ID is a supported global context variable, then \$\$PER_ID can be referenced within the service script.

```
move $$PER_ID to "schema/customerId";
```

NOTE: As described in [Context Variables](#), a service script may declare context variables that use the same two dollar sign syntax.

Perform Script and Transfer Control Statements

The edit data step supports functionality analogous to the [Perform script](#) step type and the [Transfer Control](#) step type. These are both applicable only to BPA scripts. The syntax is as follows:

- For performing a script: **performScript** 'BPA Script Name' | \$BPAScriptVariable | "bpaScriptElement";
- For transfer control: **transferControl** 'BPA Script Name' | \$BPAScriptVariable | "bpaScriptElement";

In both statements, the BPA script name may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes).

NOTE: When the script named in the **performScript** statement has finished, control will be returned to the calling BPA script. When the script named in the **transferControl** statement has finished, you will not be returned to the calling script, complete control will be granted to the transferred to script.

Navigate Statement

The edit data step supports functionality analogous to the [Navigate to a page](#) step type. This is applicable only to BPA scripts.

The syntax is **navigate** 'Navigation Code' | \$variable | "xpath";

The navigation option code may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes).

In addition, the edit data step supports the ability to indicate that the dashboard should be refreshed when navigating. This is only applicable to BPA scripts.

The syntax is **navigateAndReloadDashboard** 'Navigation Code' | \$variable | "xpath";

Invoke Map Statement

The edit data step supports functionality analogous to the [Invoke map](#) step type. This is applicable only to BPA scripts.

The syntax is **invokeMap** 'Map Name' | \$variable | "xpath" using "xpath" target bpa | page | popup;

The UI map code may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes). The target values indicate where the map should be displayed as described in the [Invoke map](#) step type. If the UI map is configured to return a value, then it can be evaluated using the **\$MAP-VALUE** variable.

```
invokeMap 'UI Map' using "dataArea";

invokeMap $variableMap using "dataArea";

invokeMap "daName/mapElement" using "dataArea" target bpa;

// $MAP-VALUE is a variable returned by the invoked map.
if ("$MAP-VALUE='continue' ")
    goto 300;
else
    terminate;
end if;
```

Declare BO with BO Group

This statement is specific to BPA scripts that plan to use the base script Main BO Maintenance Processing (**F1–MainProc**) for its Generate Edit Map statements. This script expects that the data used to display in the map is within a **boGroup** tag.

The syntax is **declareBOWithBOGroup** 'BO Name' | \$variable | "xpath" as 'DynamicEditMapSchema';

The BO code may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes).

Examples:

```
declareBOWithBOGroup 'BusinessObject' as 'newMapSchema';

declareBOWithBOGroup $variableBO as 'newMapSchema';

declareBOWithBOGroup "daName/boElement" as 'newMapSchema';
```

Generate Edit Map Statements

The 'generate edit map' statements are used to dynamically generate and launch a UI edit map based on a schema definition. The schema used may be a BO schema, a BS schema, an SS schema or a DA schema. This is applicable only to BPA scripts. The generate statement will differ based on the type of schema, but the syntax is analogous.

- **generateBOEditMap** 'BO Name' | \$variable | "xpath" using "xpath" target bpa | page | popup;
- **generateBSEditMap** 'BS Name' | \$variable | "xpath" using "xpath" target bpa | page | popup;
- **generateSSEditMap** 'SS Name' | \$variable | "xpath" using "xpath" target bpa | page | popup;
- **generateDAEditMap** 'DA Name' | \$variable | "xpath" using "xpath" target bpa | page | popup;

The BO code / BS code / SS code / DA code may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes). The target values indicate where the generated map should be displayed as described in the [Invoke map](#) step type. If the UI map is configured to return a value, then it can be evaluated using the **\$MAP-VALUE** variable.

The examples use the **generateBOEditMap** but the statements are similar for the other schema types.

```
generateBOEditMap 'BO Name' using "dataArea";

generateBOEditMap $variableMap using "dataArea";

generateBOEditMap "daName/mapElement" using "dataArea" target bpa;

// $MAP-VALUE is a variable returned by the invoked map.
if (" $MAP-VALUE='continue' ")
    goto 300;
else
    terminate;
end if;
```

Terminate Statement

The edit data step supports functionality analogous to the [Terminate](#) step type.

The following is an example of a simple **terminate** step that will stop the script.

```
if ("not(parm/spouse/name) ")
    terminate;
else
    goto addSpouse;
end-if;
```

The **terminate with error** statement is only available in a service script. When using this statement, you must supply the message category and message number. You can optionally supply substitution variables in the form of an XPath expression or literal. In addition you can optionally specify an element name within a UI map to highlight as part of the error.

The following is the syntax for terminating with an error: **terminate with error (msgcat, msgnbr %1="xpath" %2='literal' element='name' | "xpath");**.

Example:

```
if ("string(customer/lastName) = $BLANK")
    terminate with error (8000, 1001 %1="customer/
lastName" %2='Last name required' element='customer/lastName');
end-if;
```

NOTE: When specifying substitution values, make sure you use the appropriate encapsulation character: a single apostrophe to surround literal strings ('literal ') and the double quote to encapsulate XPath ("xpath ").

FASTPATH: For more information on presenting errors in a UI map, please refer to [Display Errors](#).

Debugging a BPA Script

If a BPA script has height greater than zero, then selected nodes of the script's data area can be displayed at runtime. The XML data is displayed during script execution within the BPA script's display area. Specify the XPath of an XML node from any of the BPA script's data areas, between the paired characters: '%+' and '+%'.

For example, the entire contents of the schema group node named 'input', and the specific contents of the schema element named 'output/status' will be displayed in the BPA script's display area. The debug text must be entered into the BPA script's text area and not within the script's edit data field. Debug text can be declared for any explicit step of the script.

```
display input: %+input+% , and output status: %+output/status+%
```

Script Engine Version 2 Notes

Scripting using the version 2 engine requires some extra syntax to take advantage of XPath 2 functionality. In general, any variable declared will be assumed to be a string. This means, that if you intend to construct a mathematical statement then it is necessary to explicitly declare the data type of variables as integers, numbers, or dates.

NOTE: Unless otherwise noted, all XPath examples in this topic are for the Version 1 engine – which means XPath 1. Statements that function using XPath 1 will not necessarily work for XPath 2. This is especially true when executing math, see below for examples.

Date and Time Arithmetic

XPath date/time and interval data types support arithmetic operations (+, -, * etc.) and functions, which can be used for time calculations in the same way as '1 + xs:integer(value)' is used for numeric calculations.

Compare time duration:

```
if ("xs:dateTime(fn:current-dateTime()) - xs:dateTime($updateDateTimeX)
    ge xs:dayTimeDuration(concat('PT', BO/hoursBetweenStatisticsUpdate, 'H'))")
    goto 60;
end-if;
```

Compare one date to another:

```
if ("xs:date(parm/endDate) < xs:date(parm/startDate)")
    terminate with error (11108, 11507 element='endDate');
end-if;
```

Compare a date against today's date:

```
if ("xs:date(parm/startDate) <= xs:date($CURRENT-DATE)")
    terminate with error (11108, 11507 element='endDate');
end-if;
```

Calculate the end of month:

```
// covert to ISO
move "concat($year, '-', $mon2, '-01T00:00:00')" to $monthStart;

// calculate
move "xs:dateTime($monthStart) + xs:yearMonthDuration('P1M') - xs:dayTimeDuration('P0DT1S')" to $monthEnd;

// convert from ISO to OUAUF
move "concat($year, '-', $mon2, '-', substring(string($monthEnd), 9, 2), '-23.59.59')" to $endDateTime;
```

NOTE: XPath date/time/interval formats use the ISO standard, which needs to be converted to/from formats supported in the framework.

Comparing Date/Times in String Format

Any ISO-like string format for date/time preserves the YYYY MM DD HH MM SS sequence, which is zero-padded. Regardless of separators, this format will remain appropriate for comparison operations. In particular, date/time values in the framework format “YYYY-MM-DD.HH.MM.SS” can be used with “=”, “!=”, as well as “>”, “>=”, “<”, “<=” operators.

```
// retrieve framework date/time value
invokeBS 'CM-MAXMSRMT' using "CM-MAXMSRMT";
move "string(cm-MAXMSRMT/results[1]/measurementDateTime)" to $lastMsmtDT;

// construct another date/time
move "concat($year, '-01-01-00.00.00')" to $startDateTime;

// compare using string operators
if (" $lastMsmtDT >= $startDateTime")
    move "substring($lastMsmtDT, 1, 4)" to $latestMsmtYear;
```

Converting Date/Times Between Framework and ISO

Conversion of date/time from framework format to ISO is only necessary for date/time arithmetic. Comparisons can be done with the framework format directly. The only difference between the framework format and ISO date/time formats is in the separators:

Framework: “YYYY-MM-DD.HH.MM.SS”

ISO: “YYYY-MM-DDTHH:MM:SS”

Example of converting from the framework format to ISO:

```
move "concat(substring($ouafDT, 1, 10), 'T', translate(substring($ouafDT, 12), '.', ':'))" to $isoDT;
```

Example of converting from ISO to the framework format:

```
move "concat(substring($isoDT, 1, 10), '.', translate(substring($isoDT, 12), ':', '.'))" to $ouafDT;
```

Round Money With a Dynamic Currency Scale

Because different currencies support a different number of decimals, the framework provides an API for rounding a monetary amount based on a given currency.

```
move "parm/amount" to $qty;  
move "currency/decimals" to $decimals;  
move "fn:round(xs:decimal($qty) * math:exp10(xs:double($decimals)) div math:exp10(xs:double($decimals)))"  
to "parm/roundedAmount";
```

Looping through Sequences

In XPath 2 it is possible to organize a for-loop over a sequence of integers, not only a node list.

This example shows a loop over a range of months. This is a sequence-forming construct in XPath. The XPath node list, which we are familiar with, is just another type of sequence.

```
for ($month in "1 to 12")
```

This example shows a loop over a give range of years in descending order:

```
for ($year in "fn:reverse(parm/startYear to parm/endYear)")  
  move "concat($year, '-01-01-00.00.00')" to $startDateTime;  
  move "concat($year, '-12-31-23.59.59')" to $endDateTime;  
  ...
```

This example shows a loop through a node list using ‘index’, so that other node lists can be accessed:

```
for ($idx in "1 to count(parm/touData/touList)")  
  move "parm/touData/touList[$idx]" to $tou; // access any list with this index
```

The above syntax can be used as an elegant alternative to maintaining indices separately, for example instead of the following:

```
move "0" to $idx;  
for ($item in "parm/touData/touList")  
  move "1 + xs:integer($idx)" to $idx;
```

String Padding and Decimal Formatting

This is used with specific input formats or output formatting. It is applicable to zero, space and other types of padding.

This example shows prefixing for date/time components, for example producing “2010-01-02” instead of “2010-1-2”.

```
move "substring(concat('0', string($month)), string-length(string($month)), 2)" to $mon2;
```

This example shows suffixing for adding decimal zero-padded alignment, for example producing “12.30” and “4.00” instead of “12.3” and “4”. The example performs 3 tasks: rounding to 2 decimals, inserting a period if necessary, and zero padding.

```
// round and zero-pad to 2 decimals  
move "$item/amount" to $qty;  
move "fn:round(xs:double($qty) * 100) div 100" to $qty;  
move "string($qty)" to $qty;  
move "concat(substring-before(concat($qty, '.'), '.'), '.', substring(concat(substring-  
after($qty, '.'), '00'), 1, 2))" to $qty;
```

Ternary Operation

This makes a choice between values based on a condition, so that it could be used in a single expression instead of an if/else block. It is known in C/C++ as 'cond ? value1 : value2' or in BASIC as 'IFF(cond, value1, value2)'. In XPath the syntax is: "if (cond) then value1 else value2". Note this is not the top-level scripting if-statement block.

In XPath this is an expression, which can be combined with other expressions. In scripting it can be used as:

```
move "if (string(Dl-UnitOfMeasure/  
measuresPeakQuantity) = 'DlMP') then 'DlMX' else 'DlSM' " to $func;
```

Pipeline Processing

In scripting, it is not easy to create a simple reusable piece of code as there are no local functions, and a separate script call is a coding overhead and requires packing/unpacking parameters. To avoid copying and pasting the same code block between similar script stages, consider 'pipelining', which is breaking the overall process into separate top-level steps, some of which could be shared between alternating paths. This is common for parameter preparation and output formatting. An intermediate result between stages can be stored in a "parm" substructure.

Instead of this code:

```
if ("type = A")  
  prepare params ...  
  call services for A ...  
  format output ...  
end-if;  
if ("type = B")  
  prepare params ...  
  call services for B ...  
  format output ...  
end-if;
```

Consider this alternative:

```
prepare params ...  
  
if ("type = A")  
  call services for A ...  
end-if;  
if ("type = B")  
  call services for B ...  
end-if;  
  
format output ...
```

XPath 2 Functions

Script engine version 2 supports XQuery 1.0 Functions and Operators, and XQuery 1.0 standard itself with some minor limitations. Below are the URLs to both specifications. The first link has the functions/operators available to use from XQuery.

- <http://www.w3.org/TR/xpath-functions/>
- <http://www.w3.org/TR/xquery/>

The following can only access local file systems. (For other protocols like http they will return an empty sequence):

- fn:doc
- fn:collection

How To Set Up Go To Steps

Go to steps allow you to jump to a step other than the next step. Additional fields required for **Go To** steps:

Next Step defines the step to which the script should jump.

How To Set Up Invoke Business Object Steps

Invoke business object steps allow you to interact with a [business object](#) in order to obtain or maintain its information.

The following additional fields are required for **Invoke business object** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the [data area](#) to be passed to and from the server when communicating with the **Business Object**. Indicate the **Action** to be performed on the object when invoked. Valid values are **Add, Delete, Fast Add (No Read), Fast Update (No Read), Read, Replace, Update**.

NOTE: Performance note. The actions **Fast Add** and **Fast Update** should be used when the business object's data area does not need to be re-read subsequent to the **Add** or **Update** action. In other words, the **Add** and **Update** actions are equivalent to **Fast Add + Read** and **Fast Update + Read**.

The business object call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in [global variables](#). This field is only applicable to BPA scripts.

NOTE: Error technique. Let's assume a scenario where a business object is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

How To Set Up Invoke Business Service Steps

Invoke business service steps allow you to interact with a [business service](#).

The following additional fields are required for **Invoke business service** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the [data area](#) to be passed to and from the server when the **Business Service** is invoked.

The business service call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in [global variables](#). This field is only applicable to BPA scripts.

NOTE: Error technique. Let's assume a scenario where a business service is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

How To Set Up Invoke Service Script Steps

Invoke service script steps allow you to execute a [service script](#).

The following additional fields are required for **Invoke service script** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the [data area](#) to be passed to and from the server when the **Service Script** is invoked.

The service script call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in [global variables](#). This field is only applicable to BPA scripts.

NOTE: Error technique. Let's assume a scenario where a service script is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

How To Set Up Label Steps

Label steps allow you to describe what the next step(s) are doing. Steps of this type are helpful to the script administrators when reviewing or modifying the steps in a script, especially when a script has many steps. When designing a script, the label steps enable you to provide a heading for common steps that belong together. The script tree displays steps of this type in a different color (green) so that they stand out from other steps.

There are no additional fields for **Label** steps.

How To Set Up Move Data Steps

Move data steps allow you to move data (from a source to a destination). The following additional fields are required for **Move data** steps:

Source Field Type, **Source Field Name** and **Source Field Value** define what you're moving. The following points describe each field type:

- **Context Variable.** Use this field type in a plug-in or service script if the source value is a variable initiated in a higher level script. This is only applicable to Service Scripts and Plug-in Scripts.
- **Current To Do Information.** Use this field type when the source value resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**. This is only applicable to BPA Scripts.
- **Data Area.** Use this field type when the field being compared is one that you put into one of the script's data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Global Context.** Use this field type when the source value is a [global context variable](#). This is only applicable to BPA Scripts.
- **Page Data Model.** Use this field type when the source value resides on any of the tab pages in the [object display area](#) (i.e., the source field doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.
- **Portal Context.** Use this field type when the source value is a variable in the portal context. This is only applicable to BPA Scripts.
- **Predefined Value.** Use this field type when the source value is a constant value defined in the script. When this field type is used, use **Source Field Value** to define the constant value. Refer to [How To Use Constants In Scripts](#) for instructions on how to use constants.

NOTE: Concatenating fields together. You can also use **Predefined Value** if you want to concatenate two fields together. For example, let's say you have a script that merges two persons into a single person. You might want this script to change the name of the person being merged out of existence to include the ID of the person remaining. In this example, you could enter a **Source Field Value** of **%ONAMEmerged into person %PERID** (where **ONAME** is a field in temporary storage that contains the name of the person being merged out of existence and **PERID** contains the ID of the person being kept). Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values to compose the field value.

- **Temporary Storage.** Use this field type when the source value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.

- **User Interface Field.** Use this field type when the source value resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.

Destination Field Type and **Destination Field Name** define where the source field will be moved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Context Variable.** Use this field type in your plug-in or service script if you use a variable to communicate information to a lower level service script or schema. This is not applicable to BPA Scripts.
- **Data Area.** Use this field type when the destination field resides on one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the destination field resides on any of the tab pages in the [object display area](#) (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.
- **Portal Context.** Use this field type when the destination to be updated is in the current portal context. This is only applicable to BPA Scripts.
- **Temporary Storage.** Use this field type when the destination field resides in temporary storage. Use **Field Name** to name the field in temporary storage. Use **Field Name** to name the field in temporary storage. Refer to [How To Name Temporary Storage Fields](#) for more information.
- **User Interface Field.** Use this field type when the destination field resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.

NOTE: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

How To Set Up Terminate Steps

Terminate steps cause a server-based script to end processing successfully or issue an error.

The following additional fields are required for **Terminate** steps:

Error indicates whether an error should be thrown or not. If error, **Error Data Text** must be specified, indicating the error message and any message substitution parameters. Refer to [Edit Data Syntax](#) the actual syntax of initiating an error message.

NOTE: The ability to terminate a step in error is only supported for server-based scripts.

Step Types Applicable to BPA Scripts only

The contents of this section describe step types that are only applicable to BPA scripts.

How To Set Up Display Text Steps

Display text steps cause a text string to be displayed in the script area. Steps of this type can be used to provide the user with guidance when manual actions are necessary. In addition, they can be used to provide confirmation of the completion of tasks.

The information you enter in the **Text** field is displayed in the [script area](#) when the step is executed.

The text string can contain [substitution variables](#) and [HTML formatting commands](#). Also note that for debugging purposes, you can display an entire data area (or a portion thereof) by entering `%+...+%` where ... is the name of the node whose element(s) should be displayed.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Height Steps

Height steps are used to change the height of the script area to be larger or smaller than the standard size.

The following additional fields are required for **Height** steps:

Script Window Height defines the number of **Pixels** or the **Percentage** (according to the **Height Unit**) that the script window height should be adjusted. The percentage indicates the percentage of the visible screen area that the script area uses. For example, a percentage value of **100** means that the script area will use the entire area.

NOTE: Standard Number of Pixels. The default number of pixels used by the script area is **75**.

NOTE: Adjust script height in the first step. If you want to adjust the height of the script area, it is recommendation to define the **height** step type as your first step. Otherwise, the script area will open using the standard height and then readjust, causing the screen to redisplay.

NOTE: Hide script area. You could use this type of step to set the height to **0** to hide the script area altogether. This is useful if the script does not require any prompting to the user. For example, perhaps you define a script to take a user to a page and with certain data pre-populated and that is all.

NOTE: Automatically close script area. If you want the script area to close when a script is completed, you could define the final step type with a height of 0.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Input Data Steps

Input data steps cause the user to be prompted to populate an input field in the script area. The input value can be saved in a field on a page or in temporary storage. A **Continue** button always appears adjacent to the input field. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Input Data** steps:

Destination Field Type and **Destination Field Name** define where the input field will be saved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model.** Use this field type to put the input field into a field that resides on any of the tab pages in the [object display area](#) (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage.** Use this field type to put the input field into temporary storage. Use **Field Name** to name the field in temporary storage. Refer to [How To Name Temporary Storage Fields](#) for more information.
- **User Interface Field.** Use this field type to put the input field into a field that resides on the currently displayed tab page. Note, if you want to execute underlying default logic, you must populate a **User Interface Field**. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Invoke Function Steps

NOTE: Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use a step that invokes one of the above configuration tool objects in a script rather than defining a function.

Invoke function steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class.

FASTPATH: You must set up a function before it can be referenced in a script. Refer to [Maintaining Functions](#) for the details.

The following additional fields are required for **Invoke Function** steps:

Function defines the name of the function. The function's **Long Description** is displayed below.

When a function is invoked, it will either be successful or return an error. The next two fields control the step to which control is passed given the outcome of the function call:

- **If Success, Go to** defines the step that is executed if the function is successful.
- **If Error, Go to** defines the step that is executed if the function returns on error. Refer to [How To Use Constants In Scripts](#) for a list of the global variables that are populated when a function returns an error.

NOTE: Error technique. If a function returns an error, we recommend that you invoke a step that transfers control to a script that displays the error message information and stops (note, the error information is held in [global variables](#)). You would invoke this script via a **Transfer Control**.

The **Send Fields** grid defines the fields whose values are sent to the function and whose field value source is not **Defined On The Function**. For example, if the function receives an account ID, you must define the name of the field in the script that holds the account ID.

- **Field** contains a brief description of the field sent to the function.
- **Source Field Type** and **Mapped Field / Value** define the field sent to the function. Refer to the description of Source Field under [How To Set Up Move Data Steps](#) for a description of each field type.
- **Comments** contain information about the field (this is defined on the function).

The **Receive Fields** grid defines the fields that hold the values returned from the function. For example, if the function returns an account's customer class and credit rating, you must set up two fields in this grid.

- **Field** contains a brief description of the field returned from the function.
- **Destination Field Type** and **Mapped Field** define the field returned from the function. Refer to the description of Destination Field under [How To Set Up Move Data Steps](#) for a description of each field type.
- **Comments** contain information about how the field (this is defined on the function).

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Invoke Map Steps

Invoke map steps are used to invoke a **UI Map** to display, capture and update data using an HTML form. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Invoke map** steps:

Group Name references the [data area](#) to be passed to and from the server when rendering the HTML form associated with the **Map**.

Use **Target Area** to designate where the map will be presented.

- Select **BPA Zone** if the map should be presented within the [script area](#).
- Select **Page Area** if the map should be presented in the [object display area](#), i.e. the frame typically used to house a maintenance page.
- Select **Pop-up Window** if the map should be launched in a separate window.

The **Returned Values** grid contains a row for every button defined on the map.

- **Returned Value** is the value returned when the user clicks the button.
- **Use as Default** can only be turned on for one entry in the grid. If this is turned on, this value's Next Script Step will be executed if the returned value does not match any other entry in the grid. For example, if the user closes a pop-up (rather than clicking a button), the default value will be used.
- **Next Script Step** defines the step to execute if the user clicks the button.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Mathematical Operation Steps

Mathematical operation steps allow you to perform arithmetic on fields. You can also use this type of step to add and subtract days from dates. For example, you could calculate a date 7 days in the future and then use this value as the customer's next credit review date. The following additional fields are required for **Mathematical Operation** steps:

Base Field Type and **Base Field Name** define the field on which the mathematical operation will be performed. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model.** Use this field type when the field resides on any of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage.** Use this field type when the field resides in temporary storage. You must initialize the temporary storage field with a Move Data step before performing mathematical operations on the field. Refer to [How To Set Up Move Data Steps](#) for more information.
- **User Interface Field.** Use this field type when the field resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

Math Operation controls the math function to be applied to the **Base Field**. You can specify +, -, /, and *. Note, if the base field is a date, you can only use + or -.

Math Field Type, **Math Field Name** and **Math Field Value** define the field that contains the value to be added, subtracted, divided, or multiplied. The following points describe each field type:

- **Current To Do Information.** Use this field type when the value resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the value resides on any of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Predefined Value.** Use this field type when the value is a constant. When this field type is used, use **Source Field Value** to define the constant value. Refer to [How To Use Constants In Scripts](#) for more information. Note, if you are performing arithmetic on a date, the field value must contain the number and type of **days/ months/ years**. For example, if you want to add 2 years to a date, the source field value would be **2 years**.
- **Temporary Storage.** Use this field type when the value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field.** Use this field type when the value resides in a field on the current tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Navigate To A Page Steps

Navigate to a page steps cause a new page (or tab within the existing page) to be displayed in the object display area. Steps of this type are a precursor to doing anything on the page. The following additional field is required for **Navigate to a page** steps:

Navigation Option defines the transaction, tab, access mode (add or change) and any context fields that are passed to the transaction in change mode. For example, if you want a script to navigate to Person - Characteristics for the current person being displayed in the dashboard, you must set up an appropriate navigation option. Refer to [Defining Navigation Options](#) for more information.

NOTE: Navigating to a page in update mode. Before you can navigate to a page in change mode, the page data model must contain the values to use for the navigation option's context fields. If necessary, you can move values into the page data model using a [Move Data step](#) first. For example, before you can navigate to a page in change mode with an account ID in context, you may need to move the desired account ID into the ACCT_ID field in the page data model. The actual field name(s) to use are listed as context fields on the [navigation option](#).

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Perform Script Steps

Perform script steps cause another BPA script to be performed. After the performed script completes, control is returned to the next step in the original script. You might want to think of the scripts referred to on steps of this type as "subroutines". This functionality allows you to encapsulate common logic in reusable BPA scripts that can be called from other BPA scripts. This simplifies maintenance over the long term.

The following additional field is required for **Perform script** steps:

Subscript is the name of the script that is performed.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Press A Button Steps

Press a button steps cause a button or link text to be 'pressed' in the [object display area](#), the [application toolbar](#) or the [page title area](#). For example, you could use this type of step to add a new row to a person's characteristic (and then you could use a **Move Data** step to populate the newly added row with a given char type and value). The following additional fields are required for **Press a button** steps:

Button Name is the name of the button to be pressed. This button must reside on the currently displayed tab page (or in the application toolbar or page actions toolbar). Refer to [How To Find The Name Of A Button](#) for more information.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Prompt User Steps

Prompt user steps cause the user to be presented with a menu of options. The options can be presented using either buttons or in the contents of a drop down. You can also use steps of this type to pause a script while the user checks something out (and when the user is ready to continue with the script, they are instructed to click a prompt button). The following additional fields are required for **Prompt User** steps:

Prompt Type controls if the prompt shown in the script area is in the form of **Button(s)** or a **Dropdown**. Note, if you use a **Dropdown**, a Continue button appears adjacent to the dropdown in the script area when the step executes. The user clicks the Continue button when they are ready for the script to continue.

The **Prompt Values** grid contains a row for every value that can be selected by a user. Note, if you use a **Prompt Type of Button(s)**, a separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button or in the dropdown entry. Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons or dropdown entries.
- **Use As Default** can only be turned on for one entry in the grid. If this is turned on for a dropdown entry, this value is defaulted in the grid. If this is turned on for a button, this button becomes the default (and the user should just have to press `Enter` (or `space`) rather than click on it).
- **Next Script Step** defines the step to execute if the user clicks the button or selects the dropdown value.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Set Focus To A Field Steps

Set focus to a field steps cause the cursor to be placed in a specific field on a page. A **Continue** button always appears in the script area when this type of step executes. The user may click the **Continue** button when they are ready for the script to continue. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Set focus to a field** steps:

Destination Field Name defines the field on which focus should be placed. This field must reside on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Transfer Control Steps

Transfer control steps cause the current BPA script to terminate and the control to pass to another BPA script. You might want to construct a BPA script with steps of this type when the script has several potential logic paths and you want to segregate each logic path into a separate BPA script (for ease of maintenance).

The following additional fields are required for **Transfer control** steps:

Subscript is the name of the script to which control is transferred.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

Additional Topics

The contents of this section provide additional information about steps.

How To Find The Name Of User Interface Fields

Follow these steps to find the name of a field that resides on a page:

- Navigate to the page in question.

- Right click in the body of the page (but not while the pointer is in an input field). Note, if the field in question resides in a grid, you must right click while the pointer is in the section that contains the grid (but not while the pointer is in an input field in the grid) - this is because there's a separate HTML document for each grid on a page.
- Select **View Source** from the pop-up menu to display the source HTML.
- Scroll to the Widget Info section (towards the top of the HTML document). It contains a list of all of the objects on a page. For example, the following is an example from the Account - Main page:

```

widget Info:
widget_ID , Element Type - label info - label
ENTITY_NAME, IL - $ENTITY_NAME - Name
ACCT_ID, IT - $ACCT_ID - Account ID
ACCT_CHECK_DIGIT, IL - $ACCT_CHECK_DIGIT - Account Check Digit
IM_ACCT_ID, IM - $SEARCH_FOR_ACC_LBL - Search for Account
COLL_CL_CD, HD - $COLL_CL_CD - Collection Class
SETUP_DT, IT - $SETUP_DT - Set Up Date
CURRENCY_CD, IS - $CURRENCY_CD - Currency Code
CIS_DIVISION, IS - $CIS_DIVISION - CIS Division
PROTECT_DIV_SW, CB - $CI_ACCT$PROTECT_DIV_SW - Protect CIS Division
CUST_CL_CD, IS - $CUST_CL_CD - Customer Class
ACCESS_GRP_CD, IT - $ACCESS_GRP_CD - Access Group
IM_ACCESS_GRP_CD, IM - $SRCH_ACC_GRP_CD - Search for Access Group
ACCESS_DESCR, IL - $DESCR - Description
ACCT_MGMT_GRP_CD, IT - $ACCT_MGMT_GRP_CD - Account Management Group
IM_ACCT_MGMT_GRP_CD, IM - $SEARCH_FOR_TDR_LBL - Search for To Do Role
MGMT_DESCR, IL
ALERT_INFO, IA - $ALERT_INFO - Alert Information
BILL_CYC_CD, IS - $BILL_CYC_CD - Bill Cycle
BILL_AFTER_DT, IT - $BILL_AFTER_DT - Bill After
PROTECT_CYC_SW, CB - $PROTECT_CYC_SW - Protect Bill Cycle
BILL_PRT_INTERCEPT, IT - $BILL_PRT_INTERCEPT - Bill Print Intercept
IM_BILL_PRT_INTERCEPT, IM - $FOR_BILL_PRINT_LBL - Search for User
MAILING_PREM_ID, IT - $MAILING_PREM_ID - Mailing Premise
IM_MAILING_PREM_ID, IM - $SEARCH_FOR_MAI_LBL - Search for Mailing Premise
PREM_INFO, IL
PROTECT_PREM_SW, CB - $PROTECT_PREM_SW - Protect Mailing Premise
dataframe, GD

```

The field names that you'll reference in your scripts are defined on the left side of the HTML (e.g., ENTITY_NAME, ACCT_ID, CUST_CL_CD, etc.).

The names of fields that reside in scrolls are in a slightly different format. The following is an example of the HTML for the persons scroll that appears on Account - Person. Notice that the fields in the scroll are prefixed with the name of the scroll plus a \$ sign. For example, the person's ID is called **ACCT_PER\$PER_ID**.

```

widget Info:
widget_ID , Element Type - label info - label
ENTITY_NAME, IL - $ENTITY_NAME - Name
PREM_INFO, HD
ACCT_ID, IT - $ACCT_ID - Account ID
ACCT_CHECK_DIGIT, IL - $ACCT_CHECK_DIGIT - Account Check Digit
IM_ACCT_ID, IM - $SEARCH_FOR_ACC_LBL - Search for Account
ACCT_PER$recordCount, SN - $OF_LBL - Of
ACCT_PER$PER_ID, IT - $PER_ID - Person ID
IM_ACCT_PER$PER_ID, IM - $FOR_PERSON_LBL - Search for Person
ACCT_PER$ENTITY_NAME, IL - $ENTITY_NAME - Name
ACCT_PER$MAIN_CUST_SW, CB - $MAIN_CUST_SW - Main Customer
ACCT_PER$FIN_RESP_SW, CB - $FIN_RESP_SW - Financially Responsible
ACCT_PER$THRD_PTY_SW, CB - $THRD_PTY_SW - Third Party Guarantor
ACCT_PER$ACCT_REL_TYPE_CD, IS - $CI_ACCT_PER$ACCT_REL_TYPE_CD - Relationship Type
ACCT_PER$WEB_ACCESS_FLG, IS - $WEB_ACCESS_FLG - Web Self Service Access Flag
ACCT_PER$PFX_SFX_FLG, IS - $PFX_SFX_FLG - Prefix/Suffix
ACCT_PER$NAME_PFX_SFX, IT - $NAME_PFX_SFX - Pfx/sfx Name
ACCT_PER$RECEIVE_COPY_SW, CB - $RECEIVE_COPY_SW - Receives Copy of Bill
ACCT_PER$BILL_RTE_TYPE_CD, IS - $BILL_RTE_TYPE_CD - Bill Route Type
ACCT_PER$BILL_RTE_TYPE_INFO, IL
ACCT_PER$BILL_RTG METH_FLG, HD
ACCT_PER$BILL_FORMAT_FLG, IS - $BILL_FORMAT_FLG - Bill Format

```

The names of fields that reside in grids are in a slightly different format. The following is an example of the HTML for the names grid that appears on Person - Main. Notice that the fields in the grid are prefixed with the name of the grid plus a :x \$. For example, the person's name is called **PER_NAME:x\$ENTITY_NAME**. When you reference such a field in your script, you have the following choices:

- Substitute **x** with the row in the grid (and keep in mind, the first row in a grid is row **0** (zero); this means the second row is row **1**).
- If you want to reference the "current row" (e.g., the row in which the cursor will be placed), you can keep the **x** notation (**x** means the "current row").

```
Widget Info:
  widget_ID , Element Type - label info - label
  PER_NAME:x$NAME_TYPE_FLG, IS - $NAME_TYPE_FLG - Name Type
  PER_NAME:x$ENTITY_NAME, IT - CI_PER_NAME$ENTITY_NAME - Person Name
```

How To Find The Name Of Page Data Model Fields

You find the name of a **Page Data Model** field in the same way described under [How To Find The Name Of User Interface Fields](#). The only restriction is that you cannot refer to hidden / derived fields. However, you can refer to ANY of the object's fields regardless of the tab page on which they appear. For example, if you position the object display area to the Main tab of the Account transaction, you can reference fields that reside on all of the tab pages.

CAUTION: If you populate a **Page Data Model** field, none of the underlying default logic takes place. For example, if you populate a customer contact's contact type, none of the characteristics associated with the customer contact type are defaulted onto the customer contact. If you want the underlying defaulting to take place, you must populate a **User Interface Field**.

How To Find The Name Of A Button

If you want a **Press a button** step to press a button or click a link in the application toolbar, use one of the following names:

- IM_GOBACK
- IM_HISTORY
- IM_GOFORWARD
- IM_menuButton
- IM_USER_HOME
- IM_MY_PREF
- IM_helpButton
- IM_aboutButton

If you want a **Press a button** step to press a button in the page actions toolbar, use one of the following names:

- IM_SAVE
- IM_REFRESH
- IM_CLEAR
- IM_COPY
- IM_DELETE
- IM_ScrollBack
- IM_ScrollForward

The following buttons are also supported:

- IM_TO_DO. This brings you to the [To Do Summary](#) page.
- IM_PrevTo Do. This simulates clicking the **Previous To Do** button in the [Current To Do Zone](#).
- IM_NextTo Do. This simulates clicking the **Next To Do** button in the [Current To Do Zone](#).
- IM_CurrentTo Do. This navigates to the [To Do Entry](#) page for the user's current To Do. Refer to [A User's Current To Do](#) for more information.
- IM_MINIMIZE_DASHBOARD. Pressing this will collapse the dashboard.
- IM_MAXIMIZE_DASHBOARD. Pressing this will expand the dashboard.

Follow these steps to find the name of other buttons that reside in the object display area:

- Navigate to the page in question.
- Right click in the body of the page (but not while the pointer is in an input field). Note, if the field in question resides in a grid, you must right click while the pointer is in the section that contains the grid (but not while the pointer is in an input field in the grid) - this is because there's a separate HTML document for each grid on a page.
- The option to select may differ based on the browser you are using. For example, for some browsers, the option may be **View Source**. For others, the option may be **This Frame** and then **Frame Source**
- Scroll to the Widget Info section (towards the top of the HTML document). It contains a list of all of the objects on a page, including buttons.
- Iconized buttons (e.g., search buttons) are represented as HTML images and their field names are prefixed with **IM**. The following is an example of the HTML on the To Do Entry - Main page (notice the **IM** fields for the iconized buttons).

```
* Widget Info:
*   Widget_ID , Element Type - label info - label
*   TD_ENTRY_INFO, IL - $TD_ENTRY_INFO - To Do Info
*   TD_ENTRY_ID, IT - CI_TD_ENTRY$TD_ENTRY_ID - To Do ID
*   IM_TD_ENTRY_ID, IM - $TD_ENTRY_ID_SRCH - Search for Entry Id
*   TD_TYPE_CD, IL - $TD_TYPE_CD - To Do Type
*   TYPE_DESCR, IL
*   ROLE_ID, IL
*   ROLE_DESCR2, IL - $DESCR - Description
*   FULL_MSG, PL - $GOTO_TD_ACTION_LBL - Work on To Do
*   IM_EXP_MSG_LONG, IM - $DISPLAY_MESSAG_LBL - Display Message Explanation
```

- Transaction-specific actions buttons (e.g., the buttons use to complete or forward a To Do) are represented as switches. The following is an example of the HTML on the To Do Entry - Main page (notice the **SW** fields for the buttons). Note, if you want to **Set focus** to such a field, you would move a **Predefined Value** of **TRUE** to the switch.

```
* COMPLETE_SW, BU - $COMPLETE_SW - Complete
* FORWARD_SW, BU - $FORWARD_SW - Forward
* SEND_BACK_SW, BU - $SEND_BACK_SW - Send Back
```

How To Substitute Variables In Text

You can substitute field values into a step's text string. You do this by prefixing the field name whose value should be substituted in the string with a **%**. For example, the message, "On **%COMPLETION_DTTM** this bill was completed, it's ending balance was **%ENDING_BALANCE**" contains two substitution variables (the bill's completion date / time and the bill's ending balance).

To substitute the value of an element from a data area you need to reference its XPath location as follows: **%=XPath=**. If you want to substitute the whole XML node, not just the value, you need to reference it as follows **%+XPath+**.

Only fields linked to the [current To Do](#) and fields that reside in [temporary storage](#) and [global variables](#) can be substituted into a text string.

NOTE: You can substitute fields that reside in the User Interface or Page Data Model by first moving them into temporary storage (using a **Move data** step).

You can also substitute field values into the verbiage displayed in [prompts](#) using the same technique.

How To Use HTML Tags And Spans In Text Strings and Prompts

You can use HTML tags in a step's text string. For example, the word "Continue" will be italicized in the following text string "Press<i>Continue</i> after you've selected the customer" (the **<i>** and **</i>** are the HTML tags used to indicate that the surrounded text should be italicized).

The following are other useful HTML tags:

- **
** causes a line break in a text string. If you use **

** a blank line will appear.

- ` text ` causes the surrounded text to be colored as specified (in this case, red). You can also use hex codes rather than the color name.

Please refer to an HTML reference manual or website for more examples.

You can also use "spans" to customize the look of the contents of a text string. For example, your text string could be "Press `Continue` after you've selected the customer". This would make the word "Continue" appear as large, bold, Courier text. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

How To Use Constants In Scripts

Some steps can reference fields called **Predefined Values**. For example, if you want to compare an input value to the letter "Y", the letter **Y** would be defined as a Predefined Value's field value.

Special constants are used for fields defined as switches. When you move **TRUE** to a switch, it turns it on. When you move **FALSE** to a switch, it turns it off.

You can use a [global variable](#) as a Predefined Value. For example, if you wanted to move the current date to a field, you'd indicate you wanted to move a Predefined Value named **%CURRENT_DATE**.

How To Use Global Variables

Some explicit steps can reference fields called **Predefined Values**. In addition to referencing an ad hoc constant value (e.g., the letter **Y**), you can also reference a global variable in such a field value. A global variable is used when you want to reference system data.

Note that when using the Edit Data step type, the variable available are slightly different. Refer to [Edit Data Syntax](#) for details.

The following global variables exist for BPA scripts:

- **%PARAM-`<name>`** is the value of a parameter of that name passed in to the application when launched via the standard system URL. Refer to [Launching A Script When Starting the System](#) for more information on these parameters.
- **%PARAM-NOT-SET** is to be used to compare against **%PARAM-`<name>`** parameters to check if the parameter has been set or not when the application was launched. A parameter that has not been set would test as equal to this global variable. It is recommended to test parameters against this global variable before using them for the first time.
- **%BLANK** is a constant that contains a blank value, i.e. no value
- **%SPACE** is a constant that contains a single space value
- **%CURRENT-DATE** is the current date (as known by the browser, not the server)
- **%SYSTEM-DATE** is the server date. Note that this date is affected by the [system date override](#) logic)
- **%SAVE-REQUIRED** is a flag that contains an indication of whether the data on a page has been changed (and this requires saving). You may want to interrogate this flag to force a user to save their work before executing subsequent steps. This flag will have a value of **TRUE** or **FALSE**.
- **%NEWLINE** is a constant that contains a new line character (carriage return). Upon substitution, a line break is inserted in the resultant text.

NOTE: The constant **%NEWLINE** does not have the desired effect when the resultant text is HTML. For example, a step's text and prompt strings. This is because HTML ignores special characters such as new lines. Refer to [How To Use HTML Tags And Spans In Text](#) to learn how to cause a line break in an HTML text.

- To refer to a [global context](#) variable, use **%FIELD_NAME**. For example, if the field **SP_ID** is in the global context, you may reference **%SP_ID** to reference the ID of the service point currently in context. In addition, the following special values are supported:

- **%CONTEXT-PERSONID** is a constant that contains the ID of the current person
- **%CONTEXT-ACCOUNTID** is a constant that contains the ID of the current account
- **%CONTEXT-PREMISEID** is a constant that contains the ID of the current premise

In addition, if the script is invoking something else via one of the various “Invoke” step types and an error is returned, the following global variables contain information about the error:

- **%ERRMSG-CATEGORY** and **%ERRMSG-NUMBER** contain the unique identifier of the error message number.
- **%ERRMSG-TEXT** contains the brief description of the error.
- **%ERRMSG-LONG** contains the complete description of the error.

How To Name Temporary Storage Fields

Input Data and **Move Data** steps can create fields in temporary storage. You specify the name of the temporary storage field in the step's **Field Name**. The name of the field must NOT begin with % and must not be named the same as the [global variables](#). Besides this restriction, you can use any **Field Name** that's acceptable to JavaScript (i.e., you can name a field in temporary storage almost anything). Keep in mind that field names are case-sensitive.

How To Work With Dates

Before we discuss how to work with dates in your scripts, we need to point out that there are two types of date fields: date-only and date-time. Date-only fields only contain a date. Date-time fields contain both a date and a time. The following topics describe how to work with dates on the various step types.

NOTE: If you're working with a field that resides on the database (as opposed to a temporary storage field), the database field name will tell you what type of date it is: date-only fields are suffixed with **DT**, and date-time fields are suffixed with **DTTM**.

Move Data

If you intend to use a **Move data** step to populate a *date-time* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD-HH.MM.SS or YYYY-MM-DD. If the field is in the format YYYY-MM-DD, the time of 12:00 am will be defaulted.
- If the destination field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE_DTTM:
 - First, you populate the date portion of the field. To do this, you'd move a date (this value can be in any valid date format that a user is allowed to enter) to a field called EXPIRE_DTTM_FWDDTM_P1. In other words, you suffix **_FWDDTM_P1** to the field name.
 - If you want to populate the time, you'd move the time (again, the field value can be in any format that a user could use to enter a time) to a field called EXPIRE_DTTM_FWDDTM_P2. In other words, you suffix **_FWDDTM_P2** to the field name.

If you intend to use a **Move data** step to populate a *date-only* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD.
- If the destination field resides in the *user interface*, the source field can be in any valid date format that a user is allowed to enter.

NOTE: **%CURRENT-DATE**. Keep in mind that the [global variable](#) **%CURRENT-DATE** contains the current date and you can move this to either a page data model, user interface, or temporary storage field. If you move **%CURRENT-DATE** to a temporary storage fields, it is held in the format YYYY-MM-DD.

Mathematical Operation

If you intend to use a **Mathematical operation** step to calculate a date, you can reference both date-only and date-time fields. This is because mathematical operations are only performed against the date portion of date-time fields.

Mathematical operations are limited to adding or subtracting days, months and years to / from a date.

NOTE: A useful technique to perform date arithmetic using the current date is to move the [global variable](#) %CURRENT-DATE to a temporary storage field and then perform the math on this field.

Input Data

If you intend to use an **Input data** step on a *date-time* field, please be aware of the following:

- If the field resides in the *page data model*, the user must enter a value in the format YYYY-MM-DD-HH.MM.SS (and therefore we do not recommend doing this).
- If the field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE_DTTM:
 - First, you populate the date portion of the field. To do this, you'd input the date (this value can be in any valid date format that a user is allowed to enter) in a field called EXPIRE_DTTM_FWDDTM_P1. In other words, you suffix **_FWDDTM_P1** to the field name.
 - If you want to populate the time, you'd input the time (again, the field value can be in any format that a user could use to enter a time) in a field called EXPIRE_DTTM_FWDTTM_P2. In other words, you suffix **_FWDDTM_P2** to the field name.

If you intend to use an **Input data** step to populate a *date-only* field, please be aware of the following:

- If the field resides in the *page data model*, the user must enter a value in the format YYYY-MM-DD (and therefore we do not recommend doing this).
- If the field resides in the *user interface*, the user can enter any valid date format.

How To Use To Do Fields

As described under [Executing A Script When A To Do Entry Is Selected](#), you can set up the system to automatically launch a script when a user selects a To Do entry. These types of scripts invariably need to access data that resides on the selected To Do entry. The following points describe the type of information that resides on To Do entries:

- **Sort keys.** These values define the various ways a To Do list's entries may be sorted. For example, when you look at the bill segment error To Do List, you have the option of sorting the entries in error number order, account name order, or in customer class order. There is a sort key value for each of these options.
- **Message parameters.** These values are used when the system finds %n notation within the message text. The %n notation causes field values to be substituted into a message before it's displayed. For example, the message text **The %1 non-cash deposit for %2 expires on %3** will have the values of three fields merged into it before it is displayed to the user (%1 is the type of non-cash deposit, %2 is the name of the customer, and %3 is the expiration date of the non-cash deposit). Each of these three values is stored as a separate message parameter on the To Do entry.
- **Drill keys.** These values are the keys passed to the page if a user drilled down on the entry (and the system wasn't set up to launch a script). For example, a To Do entry that has been set up to display an account on the account maintenance page has a drill key of the respective account ID.
- **To Do ID.** Every To Do entry has a unique identifier referred to as its To Do ID.

You can access this information in the following types of steps:

- **Move Data** steps can move any of the above to any data area. For example, you might want to move a To Do entry's drill key to the page data model so it can be used to navigate to a specific page.
- **Conditional Branch** steps can perform conditional logic based on any of the above. For example, you can perform conditional logic based on a To Do entry's message number (note, message numbers are frequently held in sort keys).

- **Mathematical Operation** steps can use the above in mathematical operations.

A To Do entry's sort key values are accessed by using a **Field Type** of **Current To Do Information** and a **Field Name** of **SORTKEY[index]**. Note, you can find an entry's potential sort keys by displaying the entry's To Do type and navigating to the [Sort Keys](#) tab. If you want to reference the first sort key, use an index value of **1**. If you want to use the second sort key, use an index value of **2** (and so on).

A To Do entry's drill key values are accessed by using a **Field Type** of **Current To Do Information** and a **Field Name** of **DRILLKEY[index]**. Note, you can find an entry's potential drill keys by displaying the entry's To Do type and navigating to the [Drill Keys](#) tab. If you want to use the first drill key, use an index value of **1**. If you want to use the second drill key, use an index value of **2** (and so on).

A To Do entry's message parameters are accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **MSGPARAM[index]**. Note, because a To Do type can have an unlimited number of messages and each message can have different parameters, finding an entry's message parameters requires some digging. The easiest way to determine these values is to display the To Do entry on [To Do maintenance](#). On this page, you will find the entry's message category/number adjacent to the description. Once you know these values, display the message category/number on [Message Maintenance](#). You'll find the message typically contains one or more %n notations (one for each message parameter). For example, the message text **The %1 non-cash deposit for %2 expires on %3** has three message parameters. You then need to deduce what each of the message parameters are. You do this by comparing the message on the To Do entry with the base message (it should be fairly intuitive as to what each message parameter is). If we continue using our example, **%1** is the non-cash deposit type, **%2** is the account name, and **%3** is the expiration date. You can access these in your scripts by using appropriate index value in **MSGPARAM[index]**.

A To Do entry's unique ID is accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **TD_ENTRY_ID**.

In addition, any of the above fields can be [substituted into a text string or prompt](#). Simply prefix the To Do field name with a % as you would fields in temporary storage. For example, assume you want your script to display the following text in the script area: "ABC Supply does not have a bill cycle" (where ABC Supply is the account's name). If the first sort key linked to the To Do entry contains the account's name, you'd enter a text string of **%SORTKEY[1] does not have a bill cycle**.

How To Reference Fields In Data Areas

Various step types involve referencing field elements residing in the [script's data areas](#). To reference an element in a data area you need to provide its absolute XPath notation starting from the data area name. For example, use "CaseLogAdd/caseID" to reference a top-level "caseID" element in a script data area called "CaseLogAdd".

You don't have to type in long XPath notations. Use the **View Script Schema** hyperlink provided on the [Script - Step](#) tab page to launch the script's data areas schema.

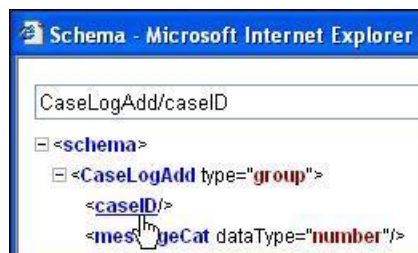


Figure 59: Schema Viewer

Doing this opens the [schema viewer](#) window where you can:

- Click on the field element you want to reference in your script step. The system automatically populates the text box on the top with the element's absolute XPath notation.
- Copy the element's XPath notation from the text box to your script.

You can also use the **View Data Area**, **View Service Script Data Area**, or **View Plug-In Script Data Area** links on [Script - Data Area](#) to the same effect. These open up the schema viewer for a specific data area respectively.

Script - Data Area

Use this page to define the data areas used to pass information to and from the server or any other data area describing your temporary storage. Open this page using **Admin > System > Script** and then navigate to the **Data Area** tab.

Description of Page

The grid contains the script's data areas declaration. For steps that invoke an object that is associated with a schema, you must declare the associated schema as a data area for your script. In addition, if you have defined one or more data areas to describe the script's temporary storage, you need to declare them too. The following bullets provide a brief description of each field on a script data area:

- **Schema Type** defines the type of schema describing the data area's element structure.
- The data area's schema is the one associated with the referenced **Object**. Only objects of the specified Schema Type may be selected.
- **Data Area Name** uniquely identifies the data area for referencing purposes. By default, the system assigns a data area with the associated object name.
- Click on the **View Data Area** link to view the data area's schema in the [schema viewer](#) window.

The **View Service Script Data Area** link appears for service scripts only. Use this link to view the script's parameters data area schema in the [schema viewer](#) window.

The **View Plug-In Script Data Area** link appears for plug-in scripts only. Use this link to view the script's parameters data area schema in the [schema viewer](#) window.

FASTPATH: Refer to [A Script May Declare Data Areas](#) for more information on data areas.

Script - Schema

Use this page to define the data elements passed to and from a service script. Open this page using **Admin > System > Script** and then navigate to the **Schema** tab.

NOTE: Conditional tab page. This tab page only appears for [service scripts](#).

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the script name and description.

The [Schema Designer](#) zone allows you to edit the service script's parameters schema. The purpose of the schema is to describe the input and output parameters used when invoking the script.

NOTE: Refer to [Schema Nodes and Attributes](#) for a complete list of the XML nodes and attributes available to you when you construct a schema.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Script - Eligibility

Use this page to define a script's eligibility rules. Open this page using **Admin > System > Script** and then navigate to the **Eligibility** tab.

NOTE: Conditional tab page. This tab page only appears for [BPA scripts](#).

Description of Page

Use the **Eligibility Option** to indicate whether the script is **Always Eligible**, **Never Eligible** or to **Apply Eligibility Criteria**. The remaining fields on the page are only visible if the option is **Apply Eligibility Criteria**.

CAUTION: The following information is not intuitive; we strongly recommend that you follow the guidelines under [The Big Picture Of Script Eligibility](#) before attempting to define this information.

The **Eligibility Criteria Group** scroll contains one entry for each group of eligibility criteria. The following fields may be defined for each group:

- Use **Sort Sequence** to control the relative order in which the group is executed when the system determines if the script should appear in the [script search](#).
- Use **Description** and **Long Description** to describe the criteria group.
- Use **If Group is True** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **True**.
 - Choose **Eligible** if this script should appear.
 - Choose **Ineligible** if this script should not appear.
 - Choose **Check Next Group** if the next criteria group should be checked.
- Use **If Group is False** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **False**.
 - Choose **Eligible** if this script should appear.
 - Choose **Ineligible** if this script should not appear.
 - Choose **Check Next Group** if the next criteria group should be checked.

The grid that follows contains the script's eligibility criteria. Think of each row as an "if statement" that can result in the related eligibility group being true or false. For example, you might have a row that indicates the script is eligible if the current account in context belongs to the residential customer class. The following bullets provide a brief description of each field on an eligibility criterion. Please refer to [Defining Logical Criteria](#) for several examples of how this information can be used.

- Use **Sort Sequence** to control the order in which the criteria are checked.
- Use **Criteria Field** to define the field to compare:
 - Choose **Algorithm** if you want to compare anything other than a characteristic. Push the adjacent search button to select the algorithm that is responsible for retrieving the comparison value. Click [here](#) to see the algorithm types available for this plug-in spot.
 - Some products may also include an option to choose **Characteristic**. Choosing this option displays adjacent fields to define the object on which the characteristic resides and the characteristic type. The objects whose characteristic values may be available to choose from depend on your product.
- Use **Criteria Comparison** to define the method of comparison:
 - Choose **Algorithm** if you want an algorithm to perform the comparison and return a value of True, False or Insufficient Data. Push the adjacent search button to select the algorithm that is responsible for performing the comparison. Click [here](#) to see the algorithm types available for this plug-in spot.
 - Choose any other option if you want to compare the **Criteria Field** using a logical operator. The following options are available:
 - Use **>**, **<**, **=**, **>=**, **<=**, **<>** (not equal) to compare the **Criteria Field** using standard logical operators. Enter the comparison value in the adjacent field.

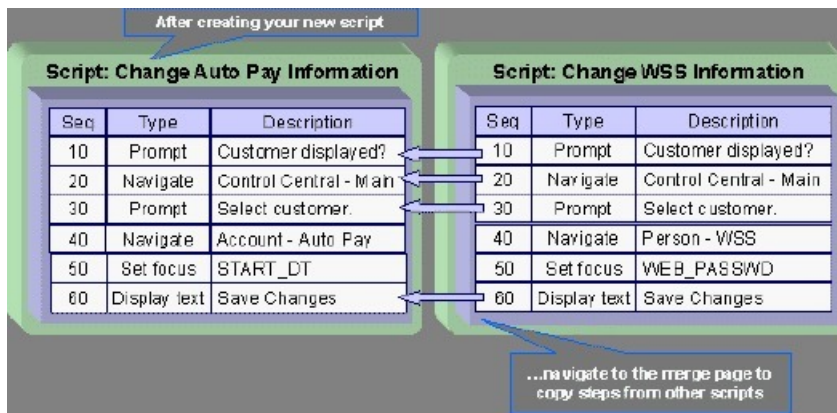
- Use **IN** to compare the **Criteria Field** to a list of values. Each value is separated by a comma. For example, if a field value must equal **1, 3** or **9**, you would enter a comparison value of **1,3,9**.
- Use **BETWEEN** to compare the **Criteria Field** to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.
- The next three fields control whether the related logical criteria cause the eligibility group to be considered true or false:
 - Use **If True** to control what happens if the related logical criterion returns a value of True. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
 - Use **If False** to control what happens if the related logical criterion returns a value of False. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
 - Use **If Insufficient Data** to control what happens if the related logical criterion returns a value of "Insufficient Data". You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.

Merging Scripts

Use the Script Merge page to modify an existing script by copying steps from other scripts. The following points summarize the many diverse functions available on the Script Merge transaction:

- You can use this transaction to renumber steps (assign them new sequence numbers).
- You can use this transaction to move a step to a different position within a script. When a step is moved, all references to the step are changed to reflect the new sequence number.
- You can use this transaction to delete a step.
- You can use this transaction to copy steps from other scripts. For example,
 - You may want to create a script that is similar to an existing script. Rather than copying all the information from the existing script and then removing the inapplicable steps, this page may be used to selectively copy steps from the existing script to the new script.
 - You may have scripts that are very similar, but still unique. You can use this transaction to build large scripts from smaller scripts. In this scenario, you may choose to create special 'mini' scripts, one for each of the various options that may make a script unique. Then, you could use the script merge page to select and merge the mini scripts that are applicable for a main script.

NOTE: The target script must exist prior to using this page. If you are creating a new script, you must first create the [Script](#) and then navigate to the merge page to copy step information.



NOTE: Duplicate versus Merge. The **Script** page itself has **duplication** capability. You would duplicate a script if you want to a) create a new script AND b) populate it with *all* the steps from an existing script.

Script Merge

Open **Admin > System > Script Merge** to open this page.

Description of Page

For **Original Script**, select the target script for merging steps.

For **Merge From Script**, select the template script from which to copy the steps.

NOTE: You may only copy steps from one Merge From script at a time. If you want to copy steps from more than one script, select the first Merge From script, copy the desired steps, save the original script, and then select the next Merge From script.

The left portion of the page displays any existing steps for the **Original Script**. The right portion of the page displays the existing steps for the **Merge From Script**.

You can use the **Copy All** button to copy all the steps from the **Merge From** script to the **Original** script. If you use **Copy All**, the steps are added to the end of the original script.

Each time you save the changes, the system renumbers the steps in the original script using the **Start From Sequence Number** and **Increment By**.

Merge Type indicates **Original** for steps that have already been saved in the original script or **Merge** for steps that have been merged, but not yet saved. The **Sequence**, **Step Type** and **Description** for each step are displayed.

The topics that follow describe how to perform common maintenance tasks:

Resequencing Steps


If you need to resequence the steps:

- Use the up and down arrows in the Original Script grid to reorder the steps.
- Make any desired changes to the **Start From Sequence Number** or **Increment By**.
- Click Save.










The steps are given new sequence numbers according to their order in the grid.

FASTPATH: Refer to **Editable Grid** in the system wide standards documentation for more information about adding records to a collection by selecting from a list and repositioning rows within a grid.










Removing a Step from Script

If you want to remove a record linked to the Original script, click the delete button, , to the left of the record.

For example, to remove the **Reset existing bundle XML** step, click the  icon.

		Merge Type	Sequence	Step Type	Description
	 	Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
	 	Original	20	Edit data	Edit data - Read the Bundle
	 	Original	30	Edit data	Edit data - Reset existing bundle XML
	 	Original	40	Edit data	Edit data - Create new bundle XML

After removal, the grid displays:

		Merge Type	Sequence	Step Type	Description
	 	Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
	 	Original	20	Edit data	Edit data - Read the Bundle
	 	Original	40	Edit data	Edit data - Create new bundle XML

NOTE: You cannot delete a step that is referenced by other steps unless you also delete the referencing steps, such as **Go to step** or **Prompt** type steps. The system informs you of any missing referenced steps when you attempt to save the original script.

Adding a Step to a Script

You can move any of the steps from the Merge From script to the Original Script by clicking the left arrow adjacent to the desired step. Once a record is moved it disappears from the Merge From information and appears in the Original information with the word **Merge** in the Merge Type column.

For example, to copy the **Navigate to a page** step, click the left arrow.









Merge Type	Sequence	Step Type	Description
	10	Height	Height - 0%
	20	Move data	Move data - Copy 'APP_SVC_ID' to 'APP_SVC_ID'
	30	Move data	Move data - Copy 'USR_GRP_ID' to 'USR_GRP_ID'
	40	Navigate to a page	Navigate to a page - userGroupAppService
	50	Conditional branch	Conditional branch - Compare ACTION with UPDATE
	60	Press a button	Press a button - IM_scrollSec_add
	70	Move data	Move data - Copy 'APP_SVC_ID' to 'UGP\$APP_SVC_ID'
	80	Label	Label - End of Script

The step is moved to the left portion of the page.

Merge Type	Sequence	Step Type	Description
Merge	40	Navigate to a page	Navigate to a page - userGroupAppService
	10	Height	Height - 0%
	20	Move data	Move data - Copy 'APP_SVC_ID' to 'APP_SVC_ID'
	30	Move data	Move data - Copy 'USR_GRP_ID' to 'USR_GRP_ID'
	50	Conditional branch	Conditional branch - Compare ACTION with UPDATE
	60	Press a button	Press a button - IM_scrollSec_add
	70	Move data	Move data - Copy 'APP_SVC_ID' to 'UGP\$APP_SVC_ID'
	80	Label	Label - End of Script

NOTE: If you add a step, such as **Go to step** or **Prompt** type steps, that references other steps, you must also add the referenced steps. The step references are updated to use the new sequence numbers when you save the original script. The system informs you of any referenced steps that haven't been added when you attempt to save the original script.

Removing an Uncommitted Step from a Script

	Merge Type	Sequence	Step Type	Description
	Merge	40	Navigate to a page	Navigate to userGroup
		10	Height	Height - 0%
		20	Move data	Move data - Copy 'APP_SVC_ID' to 'APP_SVC_ID'
		30	Move data	Move data - Copy 'USR_GRP_ID' to 'USR_GRP_ID'
		50	Conditional branch	Conditional branch - Compare ACTION with UPDATE
		60	Press a button	Press a button - IM_scrollSec_add
		70	Move data	Move data - Copy 'APP_SVC_ID' to 'UGP\$APP_SVC_ID'
		80	Label	Label - End of Script

Maintaining Functions

NOTE: Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use one of the above configuration tool objects in a script rather than defining a function. The documentation has not been updated throughout this section to highlight where BS, SS or BO could be used to perform the equivalent logic.

Invoke function steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class. Doing this is much more efficient than the alternative of transferring to the account page and retrieving the customer class from the Main page.

An **Invoke function** step retrieves or updates the relevant data by executing a service (on the server). These types of steps do not refer to the service directly. Rather, they reference a "function" and the function, in turn, references the service.

NOTE: Functions are abstractions of services. A function is nothing more than meta-data defining the name of a service and how to send data to it and retrieve data from it. Functions allow you to define a scriptwriter's interface to services. They also allow you to simplify a scriptwriter's set up burden as functions can handle the movement of data into and out of the service's XML document.

The topics in this section describe how to set up a function.

NOTE: You can retrieve data from all base-package objects. If you know the name of the base-package "page" service used to inquire upon an object, you can retrieve the value of any of its fields for use in your scripts. To do this, set up a function that sends the unique identifier of the object to the service and retrieves the desired fields from it.

Function - Main

Use this page to define basic information about a function. Open this page using **Admin > System > Function**.

Description of Page

Enter a unique **Function** code and **Description** for the function.

Use the **Long Description** to describe, in detail, what the function does.

Define the **Internal Service** that the function invokes.

NOTE: In this release, only page services can be invoked.

Click the **View XML** hyperlink to view the XML document used to pass data to and from the service. Doing this causes the XML document to be displayed in the Application Viewer.

NOTE: XML document may not be viewable. If you create a new page service and do not regenerate the application viewer, you will not be able to view its XML document.

The tree summarizes the following:

- The fields sent to the service. You can use the hyperlink to transfer to the **Send Fields** tab with the corresponding field displayed.
- The fields received from the service. You can use the hyperlink to transfer to the **Receive Fields** tab with the corresponding field displayed.
- Scripts that reference the function. You can use the hyperlink to transfer to the script page.

Function - Send Fields

Use this page to add or update the fields sent to the service. Open this page using **Admin > System > Function** and then navigate to the **Send Fields** tab.

NOTE: Displaying a specific field. Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

NOTE: You're defining the service's input fields. On this tab, you define which fields are populated in the XML document that is sent to the service. Essentially, these are the service's input fields.

Description of Page

Use **Sequence** to define the order of the **Send Fields**.

Enter a unique **Function Field Name** and **Description** for each field sent to the application service. Feel free to enter **Comments** to describe how the field is used by the service.

Use **Field Value Source** to define the source of the field value in the XML document sent to the service:

- If the field's value is the same every time the function is invoked, select **Defined On The Function**. Fields of this type typically are used to support "hard-coded" input values (so that the scriptwriter doesn't have to populate the field every time they invoke the function). Enter the "hard-coded" **Field Value** in the adjacent field.
- If the field's value is supplied by the script, select **Supplied By The Invoker**. For example, if the function retrieves an account's customer class, the script would need to supply the value of the account ID (because a different account ID is passed each time the function is invoked). Turn on **Required** if the invoker must supply the field's value (it's possible to have optional input fields).

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to populate the field's value in the XML document sent to the service.

NOTE: Usability suggestion. You populate a field's value in an XML document by specifying the appropriate XPath expression for each field. Rather than referring to an XPath manual, the system can create the XPath expression for you. To do this, click the adjacent **View XML** hyperlink. This will display the XML document used to communicate with the **Service** defined on the Main page. After the XML document is displayed, click the **XPath** hyperlink adjacent to

the desired field to see how the XPath expression looks. You can then cut / paste this XPath expression into the **XML Population Logic Field**.

Function - Receive Fields

Use this page to add or update the fields received from the service. Open this page using **Admin > System > Function** and then navigate to the **Receive Fields** tab.

NOTE: Displaying a specific field. Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

NOTE: You're defining the application service's output fields. On this tab, you define which fields are populated in the XML document that is received from the service. Essentially, these are the service's output fields.

Description of Page

Use **Sequence** to define the order of the **Receive Fields**.

Enter a unique **Function Field Name** and **Description** for each field received from the service. Feel free to enter **Comments** to describe the potential values returned from the service.

Turn on **Required** if the invoker must use the field.

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to retrieve the field's value from the XML document received from the service.

NOTE: Usability suggestion. You retrieve a field's value in an XML document by specifying the appropriate XPath expression for the field. Rather than referring to an XPath manual, the system can create the XPath expression for you. To do this, click the adjacent **View XML** hyperlink. This will display the XML document used to communicate with the **Service** defined on the Main page. After the XML document is displayed, click the **XPath** hyperlink adjacent to the desired field to see how the XPath expression looks. You can then copy / paste this XPath expression into the **XML Population Logic Field**.

NOTE: Fields in multiple lists. Note that the XPath expression generated in the application viewer refers to lists using a generic "list" reference. If a field within the list is unique across the service, the generic list reference is sufficient for the XML population logic. However, if the field you are trying to reference is in multiple lists, the XPath must include the list name. Adjust the Application Viewer's generated XPath by adding the list name, which can be found in the overview panel in the Service XML viewer. For example, instead of `/pageBody/list/listBody/field[@name='FIELD_NAME']`, the XPath Population Logic must read `/pageBody/list[@name='LIST_NAME']/listBody/field[@name='FIELD_NAME']`.

Attachments

Some implementations may require that attachments be available from the application. These attachments can be stored in the Attachment table and then linked to other objects if applicable.

Attachment Overview

The following topics provide additional information regarding attachment functionality.

Attachment Types

The system supports several different attachment content types, for example:

- PDF Document
- Excel Spreadsheet
- Jpeg Image
- Text Document

The attachment data itself may be text or binary. When storing the data in the application however, it is stored as text information only. As a result, the upload of an attachment that is a binary type requires a conversion prior to storing the data. When viewing the attachment, the data is converted again for display.

Each type of attachment is defined using an attachment business object. The business object includes configuration defining the supported file extensions, whether the data is binary or not and the content type that represents the type of data for the attachment.

NOTE: To view the attachment business objects provided with the base product, navigate using **Admin > Business Object > Search** and search for business objects related to the Maintenance Object for Attachments (**F1-ATCHMT**).

Owned Attachments

Attachments can be either ‘owned’ or ‘common’. An owned attachment is one that is related to a specific record. For example, the specific test results for a given device can be uploaded and linked to that device or to its test records. These types of attachments are typically uploaded and maintained via the object that owns it.

Common Attachments

Common attachments are ones that are uploaded independent of any transaction in the system. They can be used for general system or company information. Or they can be linked to more than one transaction. For example, instructions for performing a certain type of task can be uploaded as an attachment and linked to a task type where those instructions are relevant. These types of attachments are uploaded and maintained in the central Attachment portal. Objects that may refer to the attachments may link the attachments via characteristics or some other appropriate mechanism.

Emailing Attachments

The system supports a business service that may be used by system processing to send an email. The business service **F1-EmailService** supports receiving the IDs of one or more attachments as input parameters.

Refer to [Sending Email](#) for more information.

Configuring Your System for Attachments

In order to link attachments to objects in the system, there may be some configuration or implementation required to support the link. It is possible that one or more objects in your product already support attachments out of the box. Consult the product documentation for the specific object for confirmation. For objects in the system that do not support attachments out of the box, the following sections provide some guidelines for enabling support for attachments. Contact product support for more information.

Supporting Common Attachments

The attachments themselves are created / uploaded using the attachment portal. Refer to [Maintaining Attachments](#) for more information.

If your implementation has a use case where one or more common attachments may be linked to an object (and the object does not already support this functionality), the object may need to be extended to capture the attachments.

- If the object includes a characteristic collection, this is a recommended way to capture attachments. A characteristic type should be defined for each type of attachment. The characteristic type should be a foreign key type and should reference the Attachment FK reference. The characteristic entity collection should include the object that the common attachment will be linked to.
- Most characteristic collections are sequence based characteristics and would support multiple entries for the same characteristic types, if multiple attachments are applicable.
- If the object to support the attachments is governed by a business object, the implementation must extend the business object to define one or more appropriate elements used to capture the attachments. If only one attachment of a certain type is allowed, a single flattened characteristic may be used. If multiple attachments of a certain type are allowed, the BO schema may define a “flattened list” exposing the sequence and the characteristic type.
- If the object is maintained on a “fixed page” with a generic characteristic collection, no additional configuration is needed to allow users to link attachments to that object.

Supporting Owned Attachments

When creating an attachment for a specific record, the attachment itself captures the information about the related record, namely its maintenance object code and its primary key. For these types of attachments, no configuration is needed on the related business object to capture the attachments, as was the case with common attachments.

However, it is recommended to configure the user interface of the related object so that the owned attachments can be viewed and maintained from that page. This typically entails the creation of a special zone that retrieves a list of existing attachment records that reference the current record as its foreign key (owner). The zone would include links or buttons to add, upload or view an attachment.

If your product already has support for viewing and maintaining owned attachments on one of the base portals, that may be used as an example to follow. If your product does not have support for owned attachments, contact customer support for more information.

Defining a New Attachment Type

As mentioned, the product provides support for several content types. If your implementation needs to support attachments for a content type not currently supported, create a new business object copying the configuration of an existing attachment business object.

Configure the following option types for the BO:

- **Binary** indicates whether the attachment data must be converted from binary format. Binary attachments are stored in the database as text, and are then converted back to the original format when retrieved.
- **Content Type** represents the browser's mime type of the attachment.
- **Supported File Extension** specifies the valid file extensions for the content type.

Once the business object is defined, it is ready for use.

Maintaining Attachments

This section describes the functionality supported for viewing and maintaining attachments.

Navigate using **Admin > General > Attachment**. You are brought to a query portal with options for searching for common attachments.

Once an attachment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

NOTE: The base search options for the attachments query only support searching for common attachments. Owned attachments may also be viewed on the attachment maintenance portal, but a user may only drill into the attachment maintenance from the maintenance portal of the “owning” entity.

The Attachment zone provides basic information about an attachment, including the ability to upload the file and to view an uploaded file.

Adding Attachments

Common attachments may be added from the attachments portal (or via the standard menu path). In addition, your product may support attachments associated with specific records (“entity owned attachments”) which may also provide the capability to add attachments.

In both cases, when adding an attachment, you are prompted for the file to upload. Once the file is chose, the system determines the appropriate business object to associate with the attachment based on the file extension. Typically one and only one business object is found at which point you are prompted to provide the Attachment Name. (Your specific product may also require additional information at this time). Fill in the details and save.

Please note the following:

- If no business object is found for the uploaded file’s file type, an error is issued. This type of file is not currently supported as an attachment.
- If multiple business objects are found, the user must choose the appropriate one. This should be rare.

Application Viewer

The Application Viewer allows you to explore meta-data driven relationships and other deliverable files online.

NOTE: Running Stand-Alone. You can also launch the Application Viewer as a stand-alone application (i.e., you do not need to start it from within the system). Refer to [Application Viewer Stand-Alone Operation](#) for more information about running the Application Viewer as a stand-alone application.

To open the application viewer from within your application, navigate to **Admin > Implementation Tools > Application Viewer**. The application viewer may also be launched from other locations for example when viewing a section of the online help files that contain hypertext for a table name, clicking on that hypertext brings you to the definition of that table in the data dictionary.

Application Viewer Toolbar

The Toolbar provides the main controls for using the Application Viewer. Each button is described below.

Data Dictionary Button



The **Data Dictionary** button switches to the Data Dictionary application.

Physical and Logical Buttons



The **Physical** button changes the display in the List Panel from a logical name view to a physical name view. Note that the Tables are subsequently sorted by the physical name and therefore may not be in the same order as the logical name view. Once clicked, this button toggles to the Logical button.

The **Logical** button changes the display in the List Panel from a physical name view to a logical name view. Note that the Tables are subsequently sorted by the logical name and therefore may not be in the same order as the physical name view. Once clicked, this button toggles to the Physical button.

These buttons are only available in the Data Dictionary.

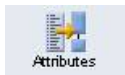
Collapse Button



The **Collapse** button closes any expanded components on the list panel so that the child items are no longer displayed.

This button is only available in the Data Dictionary viewer.

Attributes and Schema Button



The **Attributes** button changes the display in the Detail Panel from a related tables view to an attribute view. Once clicked, this button toggles to the Schema button.



The **Schema** button changes the display in the Detail Panel from an attribute view to a related tables view. Once clicked, this button toggles to the Attributes button. Note that only tables have this view available. Columns are always displayed in an attribute view.

These buttons are only available in the Data Dictionary.

Maintenance Object Button



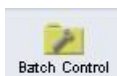
The **Maintenance Object** button switches to the Maintenance Object viewer application.

Algorithm Button



The **Algorithm** button switches to the Algorithm viewer application.

Batch Control Button



The **Batch Control** button switches to the Batch Control viewer application.

To Do Type Button



The **To Do Type** button switches to the To Do Type viewer application.

Description and Code Buttons



The **Description** button changes the display in the List Panel to Description (Code) from Code (Description). Note that the list is subsequently sorted by the description. Once clicked, this button toggles to the Code button.

The **Code** button changes the display in the List Panel to Code (Description) from Description (Code). Note that the list is subsequently sorted by the Code. Once clicked, this button toggles to the Description button.

These buttons are only available in the Batch Control and To Do Type viewers.

Service XML Button



The **Service XML** button switches to the Service XML viewer. This button is not available when you are already in the Service XML viewer.

You are prompted to enter the name of the service XML file you want to view. The name of the service XML file should be entered without the extension.

A dialog box with a light blue border. It has a label "Service XML Name" followed by a white text input field. Below the input field are two yellow buttons: "Load" on the left and "Cancel" on the right.

Select Service Button



The **Select Service** button loads another service XML file that you specify. This button is only available in the Service XML viewer.

You are prompted to enter the name of the service XML file you want to view. The name of the service XML file should be entered without the extension.

Java Docs Button



The **Java Docs** button switches to the Java Docs viewer.

Classic Button



This button is only available in the Java Docs viewer.

The **Classic** button launches the classic Javadocs viewer on a separate window. If you are more comfortable with that look you can use this viewer instead.

Preferences Button



The **Preferences** button allows you to set optional switches used by the Application Viewer. Refer to [Application Viewer Preferences](#) for more information.

Help Button



The **Help** button opens the Application Viewer help system. You used this button to access this information.

About Button



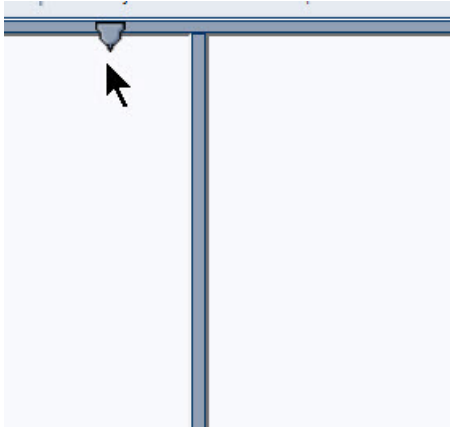
The **About** button opens a window that shows when was each Application Viewer data component recently built.

Data for all application viewer components may be regenerated to incorporate up-to-date implementation-specific information. Refer to [Application Viewer Generation](#) for further details.

Slider Icon



This "slider" icon allows you to resize the list panel and detail panel to your preferred proportions.



Data Dictionary

The data dictionary is an interactive tool that allows you to browse the database schema and to graphically view relationships between tables in the system.

To open the data dictionary, click the [Data Dictionary button](#). You can also open the data dictionary by clicking the name of a table in other parts of the application viewer or in the online help documentation.

NOTE: Data Is Generated. A background process generated the data dictionary information. Refer to [Application Viewer Generation](#) for further details.

Using the Data Dictionary List Panel

The list panel displays a list of tables and their columns. The list panel can list the table names by either their logical names or their physical names. Click the appropriate [button](#) on the toolbar to switch between the two views. The list is displayed in alphabetical order, so the order may not be the same in both views. Both views function in a similar manner.

In the list panel, you can navigate using the following options:

- Click the right arrow icon to expand a table to show its columns.
- Click the down arrow icon to collapse the column list for a table. Optionally, collapse all column lists by using the **Collapse** button.
- Click the column name to display information about the column in the detail panel.
- If the detail panel is in [related table](#) view, click the table name to view its related tables. If the detail panel is in [table detail](#) view, click the table name to display its information.

Primary And Foreign Keys

The columns in the list panel may display key information as well as the column name:

- A yellow key indicates that the column is a primary key for the table.
- A light blue key indicates that the column is a foreign key to another table. If you hover the cursor over the icon, the tool tip indicates the foreign table.
- A dark blue key indicates that the column is a conditional foreign key. A conditional foreign key represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.
- A red key indicates that the column is a logical key field. A logical key represents an alternate unique identifier of a record based on a different set of fields than the primary key.

If you hover your cursor over an icon, the tool tip indicates the key type.

Field Descriptions Shown

The language-specific, logical name of each field is shown adjacent to the physical column name in the data dictionary. You can enter an override label for a [table / field's](#) to be used throughout the system as the field's logical name. Here too it is the override label that is shown.

NOTE: Regenerate. You should regenerate the data dictionary after overriding labels. Refer to [Application Viewer Generation](#) for further details.

Using the Data Dictionary Detail Panel

The Data Dictionary detail panel displays the details of the selected item. There are three main displays for the Detail Panel:

- Related tables view
- Table detail view
- Column detail view

Related Tables View

The Related Tables view displays information about the table's parent tables and child tables. Click the [Schema](#) button in the toolbar to switch to related tables view.

In the related tables view, you can navigate using the following options:


- Click the left arrow and right arrow icons to view the related tables for that linked table. The List Panel is automatically positioned to the selected table.
- Click the maintenance object icon () to view the table's maintenance object.
- If you want to position the [List Panel](#) to view the columns for different table click the name of the table for which you want to view the columns.

Table Detail View

The table detail view displays information about the selected table. Click [Attributes](#) (in the toolbar) to switch to the table detail view.

In the table detail view, you can navigate using the following options:

- If user documentation is available for the table, click the [View User Documentation](#) link to read the user documentation that describes the table's maintenance object.
- If the table has an associated Language Table, click the link to view the Language Table details.
- If there is an associated Maintenance Program, click the link to view the source code for the maintenance program (you are transferred to the [Java Docs Viewer](#)).
- If there is an associated Key Table, click the link to view the Key Table details.

Column Detail View

Click on a column name in the list panel to switch to the column detail view. The Column Detail view displays information about the selected column.

In the column detail view, you can navigate using the following options:

- If user documentation is available for the column, click the [View User Documentation](#) link to read about the column's related maintenance object.
- If the column is a foreign key, click the table name to switch to the Table Detail view for that table.
- If the column has a Value List available (normally only present for a subset of flag and switch fields), click the link to view the source code for the copybook (you are transferred to the [Java Docs Viewer](#)).

Lookup Values

If the selected column is a lookup field its valid values are also listed. Notice that you can enter an override description for [lookup values](#). In this case the override description is shown.

NOTE: Regenerate. You should regenerate the data dictionary after overriding lookup value descriptions. Refer to [Application Viewer Generation](#) for further details.

Maintenance Object Viewer

The maintenance object viewer is an interactive tool that allows you to view a schematic diagram of a maintenance object. A maintenance object is a group of tables that are maintained as a unit.

To open the Maintenance Object Viewer, click the [Maint. Object](#) button in the application viewer or click a [maintenance object icon](#) in the Data Dictionary.

NOTE: Data Is Generated. A background process generated the maintenance object information. Refer to [Application Viewer Generation](#) for further details.


Using the Maintenance Object List Panel

The list panel displays a list of maintenance objects. In the list panel, you can click the maintenance object name to display information about the maintenance object in the detail panel.

Using the Maintenance Object Detail Panel

The Maintenance Object detail panel displays a schematic of the selected maintenance object.

In the detail panel, you can navigate using the following options:

- Click a table name to transfer to the Data Dictionary [table detail view](#) for a table. (Click the [Maint. Object](#) button in the toolbar to return to the maintenance object.)
- Click the service XML icon () to view the XML file of the Service Program used to maintain the displayed object. (Click the [Maintenance Object](#) button in the toolbar to return to the maintenance object.)

Algorithm Viewer

The algorithm viewer is an interactive tool that allows you to view algorithm types (grouped by their plug-in spot) and their related algorithms.



To open the Algorithm Viewer, click the [Algorithm](#) button in the application viewer. The Algorithm viewer may also be opened from certain locations in the online help documentation.

NOTE: Data Is Generated. A background process generates algorithm information. Refer to [Application Viewer Generation](#) for further details.

Using the Algorithm Viewer List Panel

The list panel displays a list of algorithm types and their related algorithms, grouped by their plug-in spot.

In the list panel, you can navigate using the following options:

- Click the algorithm plug-in spot description to display information about the plug-in spot in the detail panel.
- Click the right pointer  icon to expand a plug-in spot and view its algorithm types and their related algorithms.
- Click the down pointer  icon to collapse the list of algorithm types for a plug-in spot.
- Click the algorithm type name to display information about the algorithm type in the detail panel.
- Click the algorithm name to display information about the algorithm in the detail panel.

Using the Algorithm Plug-In Spot Detail Panel

The Algorithm plug-in spot detail panel displays further information about the selected plug-in spot.

Using the Algorithm Type Detail Panel

The Algorithm Type detail panel displays further information about the selected algorithm type.

In the Algorithm Type detail panel, you can navigate using the following options:

- Click on the program name to view its source in the Java docs viewer.

Using the Algorithm Detail Panel

The Algorithm detail panel displays further information about the selected algorithm.

Batch Control Viewer

The batch control viewer is an interactive tool that allows you to view batch controls.

To open the Batch Control Viewer, click the [Batch Control](#) button in the application viewer. The Batch Control viewer may also be opened from certain locations in the online help documentation.

NOTE: Data Is Generated. A background process generates batch control information. Refer to [Application Viewer Generation](#) for further details.

Using the Batch Control Viewer List Panel

The list panel displays a list of batch controls. The list panel can display the list of batch controls sorted by their code or sorted by their description. Click the appropriate [button](#) on the toolbar to switch between sorting by the code and description.

In the list panel, you can click the batch control to display information about the batch control in the detail panel.

NOTE: Not All Batch Controls Included. Note that the insertion and key generation programs for conversion (CIPV*) are not included.

Using the Batch Control Detail Panel

The batch control detail panel displays further information about the selected batch control.

In the batch control detail panel, you can navigate using the following options:

- Click on the program name to view its source in the Java docs viewer.
- If a To Do type references this batch control as its creation or routing process, click on the To Do type to view its detail in the To Do type viewer.

To Do Type Viewer

The to do type viewer is an interactive tool that allows you to view to do types defined in the system.

To open the To Do Type Viewer, click the [To Do Type](#) button in the application viewer. The To Do Type viewer may also be opened from certain locations in the online help documentation.

NOTE: Data Is Generated. A background process generates To Do type information. Refer to [Application Viewer Generation](#) for further details.

Using the To Do Type Viewer List Panel

The list panel displays a list of To Do types. The list panel can display the list of To Do types sorted by their code or sorted by their description. Click the appropriate [button](#) on the toolbar to switch between sorting by the code and description.

In the list panel, you can click the To Do type to display information about the To Do type in the detail panel.

Using the To Do Type Detail Panel

The To Do type detail panel displays further information about the selected To Do type.

In the To Do type detail panel, you can navigate using the following options:

- If the To Do type references a creation process or a routing process, click on the batch process to view its detail in the batch control viewer.
- Click on the table listed in the drill key section to view its detail in the data dictionary.
- Click on the field(s) listed in the drill key section to view its detail in the data dictionary.

Service XML Viewer

The service XML viewer is an interactive tool that allows you to browse the XML files of service programs that execute on the application server.

You can access the service XML viewer as follows:

- The maintenance object viewer allows you to view the XML file of the maintenance object's service program. This feature is implemented by viewing the maintenance object and then clicking on the [Service XML icon](#).
- When viewing a maintenance object on the [Maintenance Object](#) page, clicking the **View XML** hyperlink causes the service's XML document to be displayed in the Service XML Viewer.
- When viewing a business service on the [Business Service](#) page, clicking the **View XML** hyperlink causes the service's XML document to be displayed in the Service XML Viewer.
- When setting up a [Function](#), you may want to view the XML document used to pass data to and from the service. Clicking the **View XML** hyperlink causes the XML document to be displayed in the Service XML Viewer.

Using the Service XML Viewer Overview Panel

The overview panel displays a high level nodes and list names structure of the XML document.

In the overview panel, you can click on any node item to position the detail panel to view that item.

Using the Service XML Viewer Detail Panel

The detail panel displays nodes and attributes of the selected XML file.

Click the **xpath** button to view the XML path that should be used to reference the selected node in the XML document. The box at the top of the overview panel changes to display this information.

NOTE: Fields in multiple lists. Note that the generated XPath expression refers to lists using a generic “list” reference. For example: `/pageBody/list/listBody/field[@name='FIELD_NAME']`. If a service has a field that appears in more than one list, the above XPath may not be sufficient for referencing that field. In this case, references to the XPath should be adjusted to include the list name. The list name is visible in the overview panel. To add the list name, use `[@name='LIST_NAME']`. For example: `/pageBody/list[@name='LIST_NAME']/listBody/field[@name='FIELD_NAME']`.

Java Docs Viewer

The Java Docs viewer is an interactive tool that allows you to browse Java documentation files (Javadocs) for Java classes that execute on the application server.

NOTE: Proprietary Java Classes. A small number of Java classes have been suppressed due to their proprietary nature.

NOTE: Classic view. If you are more comfortable using the classic Javadocs viewer you may use the [Classic](#) button.

To open the Java Docs viewer from within the application viewer, click the [Java Docs button](#). Additionally, the [algorithm viewer](#) and the [batch control viewer](#) allows you to view the Javadocs of a program written in Java.

Using the Java Docs Viewer List Panel

The list panel displays a tree of Java packages where each package may be expanded to list the Java interfaces classes it includes.

In the list panel, you can navigate using the following options:

- Click the right arrow icon to expand a Java package to view the Java interfaces and classes it includes.
- Click the down arrow icon to collapse the list for a Java package. Optionally, collapse all lists by using the **Collapse** button.
- Click the Java interface or class name to display information about it in the detail panel.

The list details panel designates the interfaces and the classes as follows:

- A green dot indicates Java interfaces.
- A blue key indicates Java classes.

If you hover the cursor over the icon, the tool tip indicates whether it's an interface or a class.

Using the Java Package Detail Panel

The package detail panel displays a summary of the various Java classes that are included in the selected Java package.

Click the Java class name to display information about the Java class in the detail panel.

Using the Java Interface / Class Detail Panel

The detail panel displays Java documentation information about the selected Java interface or class.

You can navigate using hyperlinks to other locations in the current detail panel or to view the details of other Java interfaces / classes.

Application Viewer Preferences

This panel displays the Available Languages and allows you to select the language in which the labels and buttons are displayed. Select your desired language and click OK.

Application Viewer Stand-Alone Operation

You can run the Application Viewer as a stand-alone application (i.e., you do not need to launch it from the online application environment). To run it as a stand-alone application, you should copy the Application Viewer files (all files in the appViewer directory) and the online help files (all files in the help directory) to the server on which you want to run the Application Viewer.

NOTE: Online Help. If you do not copy the online help files, online help will not be available for the Application Viewer, nor will you be able to view business descriptions of the tables' maintenance objects.

To start the application viewer in stand-alone mode, launch the appViewer.html file (located in the appViewer directory).

Stand-Alone Configuration Options

You can configure the Application Viewer for stand-alone operation by modifying options in a configuration file. The Application Viewer comes with a default configuration file called `config_default.xml` (located in the `appViewer\config` directory). Create a copy of the default configuration file and rename it to `config.xml`. Modify the options described in the following table to suit the needs of your installation.

NOTE: Default Configuration. If you do not create the `config.xml` file, the Application Viewer launches with its default (internal) configuration.

Option	Description
<code>defaultLanguage</code>	The default language used when the application viewer is started. Available values are those marked as language enabled on the language page.
<code>defaultView</code>	The default view then the application viewer is started. Available values include: - Data Dictionary
<code>dataDictionary</code>	Whether the Data Dictionary is available or not: - Y - N
<code>sourceCode</code>	This property is not being used. Simply enter 'N'.
<code>baseHelpLocation</code>	The location of the stand-alone online help in relation to the application viewer. Specify the directory structure relative to the location of the directory in which the Application Viewer files are located. Note that this is the directory in which the language subdirectories for the online help are located. The default location is: ../help
<code>appViewerHelp</code>	The default help topic that is launched when the Help button is clicked in the Application Viewer. Specify a help file and anchor that is under the appropriate language directory under the <code>baseHelpLocation</code> . The default is: Framework/Admin/91AppViewer.html#SPLINKApplication_Viewer

Example Application Viewer Configuration

The following excerpt shows an example Application Viewer configuration.

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
<option id="defaultLanguage">PTB</option>
<option id="defaultView">Data Dictionary</option>
<option id="dataDictionary">Y</option>
<option id="sourceCode">N</option>
<option id="baseHelpLocation">../help</option>
<option id="appViewerHelp">Framework/Admin/91AppViewer.html#SPLINKApplication_Viewer</option>
</configuration>
```

Application Viewer Generation

The Application Viewer is initially delivered with service XML information only.

The other components of the application viewer are generated on site.

- Use the background process **F1-AVALG** to regenerate algorithm information
- Use the background process **F1-AVBT** to regenerate batch control information.
- Use the background process **F1-AVMO** to regenerate maintenance object information
- Use the background process **F1-AVTBL** to regenerate data dictionary information.
- Use the background process **F1-AVTD** to regenerate To Do type information.

These processes have been introduced so that you can more easily incorporate your implementation-specific information into the application viewer.

To keep the information shown in the application viewer current it is important to execute these background processes after you introduce changes to the corresponding system data.

NOTE: Data Generation Is Not Incremental. Each new execution of these processes first deletes existing data (if any) in the corresponding folder before it generates new data.

NOTE: Other Extensions. Service XML may also be extended to include implementation-specific information. The base package is provided with special scripts that handle this type of extension. Refer to the Software Development Kit User Guide for further information on application viewer extensions.

NOTE: War File. If your application is installed in war file format, each generation of application viewer data rebuilds the corresponding war file. The web application server then needs to be "bounced" in order to make the newly generated data available to the application viewer. Please consult your system administrator for assistance.

NOTE: Certain Web Application Servers Are Special. WebSphere and Oracle Application web application servers require an additional step in order to make the newly generated data available to the application viewer. These web application servers require a rebuild of the application ear file and its redeployment in the web application server. This step is described in the installation document. Please consult your system administrator for further details.

Defining and Designing Reports

This section describes how to configure your third party reporting tool and how to define your reports in the system to enable users to submit reports online.

The Big Picture Of Reports

The topics in this section describe the approach for designing and defining your system reports.

Integration with BI Publisher and Business Objects Enterprise

Your DBMS, your product, and BI Publisher or Business Objects Enterprise / Crystal Reports can work together to produce reports. You may choose to use a different reporting tool, but this may not be a trivial effort. This section provides high-level information about some of the business requirements that are being solved with the reporting solution.

Reports Must Be Multi-Language

The system supports a multi-language implementation and the reporting solution for the system must also support a multi-language implementation.

- If a French-speaking user requests a given report, all labels, headings and messages are in French. If the same report is requested by an English-speaking user, all information is in English
- Different fonts may be necessary for a given report (the font for Chinese is rather different than the font for English)
- Dates and numbers must be formatted as per the user's profile in your product
- Currency must be formatted as per the currency definition in your product

In order to provide the above functionality, the third party reporting tool must do the following:

- Access the system's field and message metadata for labels, headings and messages (as different values can be defined for different languages in system's metadata)
- Retrieve the appropriate font, size, and layout based on the requested report and the user's language
- Use the currency code information in the system to format currency-oriented information
- Use the user's display profile to format date and number fields

Requesting Reports from The System

Although reports are rendered in your reporting tool, users must be able to request ad-hoc reports from within the system (assuming users have the appropriate security access).

- The prompts for the input parameters must be shown in the user's language
- Users should be able to use the standard search facilities to find parameter values
- Plug-ins can be optionally used to cross-validate input parameters
- Application security must authorize ad-hoc report requests

Overview of the Data - BI Publisher

The following diagram provides an overview of where data is stored for your reports for integration with BI Publisher.

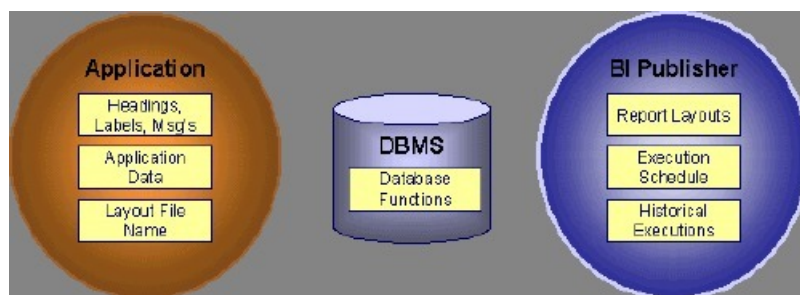


Figure 60: Application and BI Publisher

The application contains:

- The application data that appears on your reports.
- The language-specific headings, labels and messages on your reports.
- The layout file name to be used for the report.

BI Publisher contains:

- How your reports look.
- Information about scheduled reports and reports that have already run.

The DBMS contains the SQL used to retrieve the data on your reports (residing in database functions).

NOTE: BI Publisher can be configured to retrieve data via a service call. Because every business object can be read via a service call, elements that reside in an XML based column can be displayed on reports produced using BI Publisher. See your product's *Optional Products Installation Guide* for information on this configuration.

Overview of the Data - Business Objects Enterprise

The following diagram provides an overview of where data is stored for your reports for integration with Business Objects Enterprise.

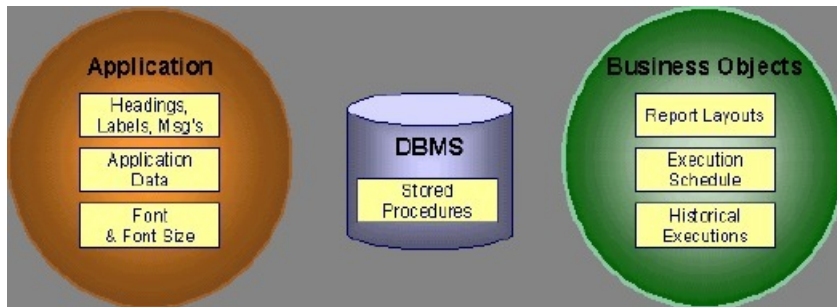


Figure 61: Application and Business Objects Enterprise

The application contains:

- The application data that appears on your reports.
- The language-specific headings, labels and messages on your reports.
- For Business Objects Enterprise, the font and size in which each report is rendered.

Business Objects Enterprise contains:

- How your reports look.
- When your reports are produced (the batch scheduler).
- Historical images of reports.

The DBMS contains the SQL used to retrieve the data on your reports (residing in stored procedures).

How To Request Reports

A user may request an ad hoc report from within your product:

- A [report submission](#) page enables a user to choose the desired report and enter the parameter values for the report
- The user must be granted security access to the report

- The request is passed to the reporting tool real time. Refer to [Configure The System to Invoke BI Publisher](#) or [Configure The System to Invoke Business Objects Enterprise](#) for more information.
- The reporting tool generates the report and displays it in a new browser window

The reporting tools' scheduler creates reports (as per your schedule)

- This function is entirely within the reporting tool. No scheduling functions reside within your product.

A user can request an ad-hoc report from within the reporting tool

- Note, the user's ID must be supplied as a parameter to the report in order for the user's profile to be used to format dates and numbers

Viewing Reports

As described above, ad-hoc reports requested from within your product are displayed immediately after they are generated in a new browser window

Crystal's report repository can be used to retrieve historical versions of a report. The [Report History](#) page allows users to open the Crystal's report execution history page and request a view of this report.

NOTE: The [Report History](#) page currently does not display historical reports for BI Publisher.

Configuring The System To Enable Reports

Configuring BI Publisher Reports

This section contains topics specific about configuring the product to interoperate with BI Publisher.

Configure the System to Invoke BI Publisher Real-time

The base product provides an [installation algorithm](#) plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For BI Publisher, the system provides an algorithm type called [F1-BIPR-INV](#), which invokes BI Publisher.

These algorithms rely on information defined in the [Reporting Options](#) table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke BI Publisher, perform the following steps:

- Create an [algorithm](#) for the appropriate algorithm type.
- On the [installation options](#), add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

Batch Scheduling in BI Publisher

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the BI Publisher software to provide the batch scheduler functionality. Refer to BI Publisher documentation for details about configuring the batch scheduler.

NOTE: The [report history](#) page currently does not display historical reports for BI Publisher.

Configuring Business Objects Enterprise Reports

This section contains topics specific about configuring the product to interoperate with Business Objects Enterprise.

Configure the System to Invoke Business Objects Enterprise Real-time

The base product provides an [installation algorithm](#) plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For Business Objects Enterprise, the system provides an algorithm type called [RPTE-INV](#), which invokes Business Objects Enterprise.

These algorithms rely on information defined in the [Reporting Options](#) table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke one of the reporting tools, perform the following steps:

- Create an [algorithm](#) for the appropriate algorithm type.
- On the [installation options](#), add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

Batch Scheduling in Business Objects Enterprise

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the Business Objects Enterprise software to provide the batch scheduler functionality. Refer to Business Objects Enterprise documentation for details about configuring the batch scheduler.

The product provides a [report history](#) page to display report instances that were produced via the batch scheduler and are stored in a repository. The report history page relies on the [reporting tool algorithm](#) to invoke Business Objects Enterprise and display the historic instances for the selected report.

Defining Reporting Options

The reporting options are provided as a mechanism for defining information needed by your reporting solution. The base product uses the reporting options to define information needed to access the reporting tool from within the system using the algorithm defined on the installation option.

Navigate to this page using **Admin > Reporting > Reporting Options**.

Description of page

The following information must be defined to interface with BI Publisher real-time. Contact your system administrator to report any problems with the settings defined here.

Reporting Folder defines the shared folder where reports are stored.

For Business Objects Enterprise, defines the name of the virtual directory on the server where Java Service pages (JSP) are located. The reporting tool algorithm uses this information to construct the URL to launch the reporting tool. The reporting tool algorithm assumes that a JSP named "logon.jsp" is located there.

Reporting Server defines the URL of the web application where the reporting tool is installed. For example, using BI Publisher, the format is: `http://<BI Publisher Server>:<port>`.

Reporting Tool User ID is not applicable when integrating with BI Publisher.

For Business Objects Enterprise, defines the user id to use when logging in.

Reporting Tool Password is not applicable when integrating with BI Publisher.

For Business Objects Enterprise, defines the password to use when logging in.

NOTE: Customize Options. The reporting options are customizable using the Lookup table. This field name is **RPT_OPT_FLG**. The reporting options provided with the system are needed to invoke the reporting tool. If your implementation requires other information to be defined as reporting options, use the lookup table to define additional values for the reporting option flag.

Where Used

This information is used by the reporting tool algorithm on the [installation option](#) to invoke the reporting tool software.

Implementations may use reporting options to record other information needed for their reporting tool.

Defining Report Definitions

For each report supplied by your installation, use the report definition page to define various attributes of the report.

Report Definition - Main

Navigate to this page using **Admin > Reporting > Report Definition**.

CAUTION: Important! If you introduce new report definitions, you must prefix the report code with **CM**. If you do not do this, there is a slight possibility that a future release of the application could introduce a new system report with the name you allocated.

Description of page

Enter an easily recognizable **Report Code** and **Description** for each report. Use the **External Reference ID** to define the identifier for this report in your external reporting tool.

Define an [application service](#) to enable users to request submission of this report online or to view report history for this report. Once you define an application service for each report, use [application security](#) to define which users may access this report.

NOTE: Access Mode. The access mode for application services related to reports must be set to **Submit/View Report**.

If you have more than one parameter defined for your report and you wish to perform cross-validation for more than one parameter, provide an appropriate **Validation Algorithm**. Click [here](#) to see the algorithm types available for this system event.

Enter a **Long Description** to more fully describe the functionality of this report. This information is displayed to the user when attempting to submit the report online or when viewing history for this report.

For BI Publisher, if you want to use one of the sample reports provided by the system, but with a different layout, indicate the layout to use for the report in the **Customer Specific Font/ Layout** field and BI Publisher uses this information instead. The name for base report layout is <report code>_Base. For example, a base layout for CI_VACANT is named CI_VACANT_Base.

For Business Objects Enterprise, the **Report Font** and **Report Font Size** are used to control the display of the report information. If you wish to use one of the sample reports provided by the system, but wish to use a different font and font size, indicate your **Customer Specific Font** in the **Customer Specific Font/Layout** field and Business Objects Enterprise uses this information instead.

Report Definition - Labels

Navigate to this page using **Admin > Reporting > Report Definition** and go to the **Labels** tab.

NOTE: Company name and logo. Note the company name used as a title in the sample reports is defined as a message on the [installation options](#). For information about installing the company logo, refer to the *Reports Configuration* chapter of the *Installation Information*.

Description of Page

In order to provide multi-language capability for each report, the labels used for the report must support multiple language definitions. For each label used by your report, indicate a unique **Sequence** and the **Field** used to define the **Label**. The label defined here should be the same label that is defined in your report layout defined in the external reporting tool.

When rendering an image of the report, the external reporting tool retrieves the appropriate label based on the language used for the report.

Report Definition - Parameters

Navigate to this page using **Admin > Reporting > Report Definition** and go to the **Parameters** tab .

Description of Page

The **Parameters** scroll contains one entry for every parameter defined for the report. To modify a parameter, simply move to a field and change its value. To add another parameter, click + to insert a row and then fill in the information for each field. The following fields display:

Parameter Code The identifier of the parameter. This must correspond to the parameter definition in the reporting tool.

Required Turn on this switch if the user must supply a value for the parameter when submitting the report.

Sort Sequence Indicate the sort sequence for this parameter. This sequence must match the parameter order defined in the reporting tool's report. It is also used when displaying the list of parameters on the [report submission](#) page.

Characteristic Type Indicate the characteristic type used to define this parameter.

Default Value Use this field to define a default value for this parameter. Default values are displayed to the user when the report is chosen on the [report submission](#) page.

Description Define a brief description of the parameter. This description is used when displaying the parameter on the [report submission](#) page.

Long Description Define a detailed description of the parameter. This description is used on the [report submission](#) page when the user requests more information for a given parameter.

Sample Reports Supplied with the Product

Depending on your specific product, there may be sample reports provided that your organization may use as they are or as a starting point for creating a [new report](#). The following sections provide an overview of the sample reports along with instructions on how to use one of the sample reports in your implementation environment.

How to Use a Sample Report Provided with the System

If you would like to use any of the sample reports, you need to perform some steps to be able to execute them in an implementation environment. This section walks you through the steps needed.

Steps Performed at Installation Time

The *Installation Guide* provides instructions for setting up and configuring your product and reporting tool to use the sample reports provided with the system. The following steps are described there.

- Setting up the stored procedures used by the sample reports.
- Defining the company title and logo used by the sample reports. Note the company name used as a title in the sample reports is defined as a message on the [installation options](#). For information about installing the company logo, refer to the *Reports Configuration* chapter of the *Installation Information*.
- Defining a user for integration with your product.
- Publishing the sample reports in BI Publisher or Business Objects Enterprise.

Contact your system administrator to verify that the above steps have occurred.

Subreports Used with Crystal Reports

The sample reports supplied with the system use several common subreports. Subreports are used in Crystal Reports to retrieve common data such as, labels and your company title. They are shared for all reports and may be reused for customer reports. Implementers may also use these subreports when designing new reports.

NOTE: Specific Subreports. This section only includes common subreports that may be reused by new reports. You may notice that other subreports are supplied with the system. These subreports provide functionality for a specific sample reports and are not meant for reuse.

Display Company Logo and Title

The subreport **CIZCOMP** receives the user id as a parameter and calls the stored procedure **CIZCOMP**. It retrieves the company's title in the user's language from the appropriate [installation message](#) record.

FASTPATH: Refer to the **Reports Configuration** chapter of the installation guide for more information about defining the location for the company logo.

Format Report Information

The subreport **CIZINST** defines shared variables that are used for formatting fields in the main report. It calls the stored procedure **CIZINST**. This subreport receives the user id and report code as parameters. It retrieves the font and font size from the [report definition](#). It retrieves the format date/time and number format from the user's [display profile](#). Finally, it retrieves the currency from the [installation](#) record and retrieves the currency symbol and position from the currency's record.

NOTE: Multi-currency. All reports support multiple currencies. Currency information is returned for each row by the appropriate stored procedure. This subreport retrieves the currency code from the Installation Options and should only be used in a report if there is no other currency information available.

Labels

The subreport **CIZLABEL** keeps all labels used in the main report. It calls the stored procedure **CIZLBALL** with the user ID as a parameter. This stored procedure returns all labels defined for all reports. The subreport selects labels specified for the current report and sets shared variables L001...L100 to store the labels. If more than 100 labels are required for a

new report, the version of the CIZLABEL subreport used for the new report should be changed to add additional shared variables.

How To Define A New Report

Use a Sample Report as a Starting Point

- Make a copy of the report and save it in an appropriate directory. Prefix the new report name with **CM**.
- Review the stored procedure(s) used for this report. Refer to the installation guide for information about where the stored procedures should be defined. If you want to change the data that is being accessed, copy the stored procedure, prefixing the new stored procedure with **CM**. Make the appropriate changes in the new version of the stored procedure. Contact your database administrator to find out the procedure for creating a new stored procedure.

NOTE: Performance considerations. When designing a stored procedure, you must consider the performance of the report when executed. Consult your database administrator when designing your database access to ensure that all issues are considered.

NOTE: Defining Messages. The stored procedures provided with the system use messages defined in message category 30. If your new stored procedures require new messages, use message category 90000 or greater, which are reserved for implementations.

- Review the parameters used by the report. Make appropriate changes to the parameters required by the report. This affects how you define your report. Refer to [Designing Parameters](#) for more information.
- Determine whether or not you require cross validation for your report parameters. If any cross validation is necessary, you should design an appropriate validation algorithm to be executed when requesting a report in your product. Refer to [Designing Validation Algorithms](#) for more information.

NOTE: Cross Validation for On-line Submission Only. The cross validation algorithm is only executed for ad-hoc report submissions via your product. If you submit this report through your reporting tool, this algorithm is not executed.

- Review the labels used by the report. Labels and other verbiage are implemented in the sample reports using a reference to the field table in the system. This enables the report to be rendered in the appropriate language for the user. For any new report label you require, you must define a new field entry. Refer to [Designing Labels](#) for more information.
- Review the layout of the report and make any desired changes based on your business needs.

When you have finished designing and coding your new report in your reporting tool, you must do the following in order for it to be usable:

- Publish the report in BI Publisher or Business Objects Enterprise. Refer to the documentation for these products for details about publishing a report. Refer to [Publishing Reports in BI Publisher](#) and [Publishing Reports in Business Objects Enterprise](#) for configuration information specific to publishing a report for integration with your product.
- Define the report. Refer to [Designing Your Report Definition](#) for more information.

Publishing Reports in BI Publisher

Please refer to the documentation for BI Publisher for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using BI Publisher.

BI Publisher Database Access

When publishing a report in BI Publisher, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the database functions related to this report in your database.

Verify BI Publisher User Access Rights

To verify the user's access rights to folders in BI Publisher:

- Open the BI Publisher Enterprise Security Center.
- Check that the role for the user has access to the appropriate report folders.

For more information, refer to the "Understanding Users and Roles" section in the Oracle Business Intelligence Publisher User's Guide.

Publishing Reports in Business Objects Enterprise

Please refer to the documentation for Business Objects Enterprise for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using Business Objects Enterprise.

Business Objects Enterprise Database Access

When publishing a report in Business Objects Enterprise, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the stored procedures related to this report in your database.

Verify Parameter Definition

This section describes how to verify parameter definitions in the Crystal Management Console (CMC).

- Once your report has been published, navigate to the CMC. This is the web-based administration component for Business Objects Enterprise and provides access to all administrative functions.
- To verify/change the settings of a report in the CMC go to the Objects management area and select the desired report by clicking its link located in the Object Title column.
- Once you have selected your report, click the **Parameters** tab to change the settings.
- If your report requires parameters to be provided by the user, you must configure the parameter settings in the CMC to ensure that parameter values are passed from the system when submitting the report via the [Report Submission](#) page. If you plan to submit reports from within your product, the **Prompt the user for new value(s) when viewing** check box should be checked.
- Click **OK** on this window and then click **Update**.

NOTE: Submitting Reports Through Business Objects Enterprise. If you plan to submit reports from Business Objects Enterprise, you must also define an appropriate initial value for each parameter, if applicable.

NOTE: User ID. The user id is defined as the first parameter in every sample report. This parameter is hidden when the report is submitted from within your product, but it must be defined in the Crystal report.

Verify Business Objects Enterprise User Access Rights

To verify the access rights for a user in CMC:

- Navigate to the **Rights** tab in the Objects management area of the CMC and check that the user has correct security level for the report.
- Integration with your product requires an access level of **View On Demand** for the user.

Designing Your Report Definition

When adding a new report, you must define it in the system to allow users to request ad-hoc reports from on-line and to take advantage of the multi-language provisions in the system. The following topics illustrate the steps to take to correctly configure your report definition.

Designing Main Report Definition Values

Refer to field description section of the [report definition](#) main page for information about defining general information about the report.

For the validation algorithm, preliminary steps are required. Refer to [Designing Validation Algorithms](#) for more information.

For the application service, preliminary steps are required. Refer to [Designing Application Services](#) for more information.

Designing Characteristic Types

The parameter tab on the report definition page uses [characteristic types](#) to define the report parameters. For each report parameter that you plan to use, you must define a characteristic type.

You do not need a unique characteristic type for each report parameter. For example, if Start Date and End Date are parameters your report, only one **Report Date** characteristic type needs to be defined. This characteristic type would be used on both date parameters.

Each characteristic type to be used as a report parameter must indicate a characteristic entity of **Report**.

To illustrate the characteristic type definitions, let's look at the sample report Tax Payables Analysis. It needs the following parameters: From Date, To Date, GL Account Type Characteristic Type and Account Type value.

NOTE: Account Type Parameters. The tax payables report must find general ledger entries that have posted to a certain distribution code. In order to find the appropriate distribution code, the report expects each distribution code to be defined with a characteristic indicating its GL account type (for example, **Revenue**, **Asset**, etc.) The report needs to know the characteristic type used to define this entry.

To support the required parameters, the following characteristic types are needed.

Char Type	Description	Type	Valid Values	Char Entities
CI_DATE	Date Parameter	Ad-hoc	(Uses validation algorithm to validate proper date entry)	Report
CI_CHTYP	Characteristic Type	FK Reference	CHAR_TYP	Report
CI_GLTY	GL Account Type	Pre-defined	A- Asset, E- Expense, LM- Liability/miscellaneous, LT- Liability/taxes, R-Revenue	Distribution Code, Report

Highlights for some of the above settings:

- We have defined a characteristic type for defining a characteristic type. This is to allow the user to indicate which Char Type on the Distribution Code is used for the GL account type. This is an FK reference type of characteristic.
- The GL account type characteristic type is referenced on both the Distribution Code entity and the report entity.

Designing Parameters

Your report definition parameters collection must define a unique parameter entry for each parameter sent to the reporting tool. The sequence of your parameters must match the sequence defined in your reporting tool.

Continuing with the Tax Payables Analysis report as an example, let's look at the parameter definitions.

Parameter Code	Description	Char Type	Default Value
P_FROM_DT	From Date	CI_DATE	N/a
P_TO_DT	To Date	CI_DATE	N/a
P_CHAR_TYPE	Account Type Characteristic	CI_CHTYP	CI_GLTYP
P_TAX_ACCTY_CHAR	Account Type Char Value for Tax Related GL Account	CI_GLTYP	LT-Liability/taxes

Highlights for some of the above settings:

- The from date and to date parameters use the same characteristic type.
- The characteristic type parameter is defined with a default value pointing to the GL account type characteristic type.
- The GL account type parameter defines the liability/taxes account type as its default value.

NOTE: User Id. The sample reports provided by the system pass the user id as the first parameter passed to the reporting tool. It does not need to be defined in the parameter collection for the report.

Designing Validation Algorithms

When designing your report definition, determine if cross validation should occur for your collection of parameters. In the Tax Payables Analysis report, there are two date parameters. Each date parameter uses the characteristic type validation algorithm to ensure that a valid date is entered. However, perhaps additional validation is needed to ensure that the start date is prior to the end date. To do this, a validation algorithm must be designed and defined on the report definition.

The system provides a sample algorithm [RPTV-DT](#) that validates that two separate date parameters do not overlap. This algorithm should be used by the Tax Payables Analysis report.

If you identify additional validation algorithm, create a new [algorithm type](#). Create an [algorithm](#) for that algorithm type with the appropriate parameter values. Plug in the new validation algorithm to the appropriate report definition.

Designing Application Services

[Application services](#) are required in order to allow a user to submit a report on-line or to view history for a report. Define an application service for each report and define the user groups that should have submit/view access to this report.

Update [report definition](#) to reference this application service.

Designing Labels

The system supports the rendering of a report in the language of the user. In order to support a report in multiple languages, the verbiage used in the report must be defined in a table that supports multiple languages. Some examples of verbiage in a report include the title, the labels and column headings and text such as "End of Report".

The system uses the [field](#) table to define its labels.

NOTE: Report Definition. This section assumes that your new report in the reporting tool has followed the standard followed in the sample reports and uses references to field names for all verbiage rather than hard-coding text in a single language.

For each label or other type of verbiage used by your report, define a field to store the text used for the verbiage.

- Navigate to the field page using **Admin > System > Field**.
- Enter a unique **Field Name**. This must correspond to the field name used in your report definition in the reporting tool and it must be prefixed with **CM**.
- Define the **Owner** as **Customer Modification**.
- Define the **Data Type** as **Character**.
- **Precision** is a required field, but is not applicable for your report fields. Enter any value here.
- Use the **Description** to define the text that should appear on the report.
- Check the **Work Field** switch. This indicates to the system that the field does not represent a field in the database.

Update the [report definition](#) to define the fields applicable for this report in the **Labels** tab.

If your installation supports multiple languages, you must define the description applicable for each supported language.

External Messages

This section describes mechanisms provided in the product that enable an implementation to configure the system to communicate with an external application.

Incoming Messages

This section provides information about support for incoming messages.

Inbound Web Services

Inbound web service functionality is provided to support receiving web service requests from an external system.

Overview

The following topics provide overview information about inbound web services (IWS).

Multiple Operations

An inbound web service supports the configuration of one or more operation per web service. Each operation defines the schema-based object to invoke to perform the desired function. An operation may refer to a Business Service, a Business Object, or a Service Script. If the IWS supports multiple operations, each operation can refer to the same or a completely different schema-based object from other operations within the IWS. In addition, each operation may define a transaction type, which is essentially an action that is supported by the schema-based object.

By default, Inbound Web Services uses the Schema Name to dictate the Request and Response for the service. The API can be overridden with custom formats by specifying Request and Response Schemas with the appropriate Request and Response XSL to transform into the relevant schema formats.

For business object based operations, when invoking the web service an action is required. This may be passed into the web service as part of the invocation or alternatively, the action may be defined when configuring the operation using the transaction type.

NOTE: Using the transaction type **Change** requires all values to be passed in. Using the transaction type **Update** allows the web service to pass only the primary key and the values to be updated. All other elements will retain their existing values.

Annotations Used for Security

When preparing to deploy inbound web services, the security aspects of the service must be decided. The product provides a default security policy that is applied when no other policy is defined: **@Policy(uri="policy:Wssp1.2-2007-Https-BasicAuth.xml", attachToWSDL=true)** which requires HTTP Basic over SSL and a WS-Security Timestamp.

If a different security policy is desired, the following options are available:

- Security policies may be attached to the Inbound Web Service via the J2EE Web Application Server. This allows for multiple policies to be attached as supported by the J2EE Web Application Server. In order to enable this capability, explicit system configuration is required so that the product does not assume the default security policy. See the subsequent bullets for more information.
- Define a system wide security policy using a feature configuration option. Find the [Feature Configuration](#) record for the **External Messages** feature type. (It may need to be defined if it does not exist). Choose the option type **Default security policy** and define an appropriate value. If your implementation wishes for the policies to be attached at the J2EE Web Application Server, define this option type with an option value of **<none>**.
- Attach a security policy to the IWS via a Web Service Annotation using the Policy annotation type (**FIPOLICY**). No base annotation is supplied by the product. If your implementation wishes for the policy of a particular IWS to be attached at the J2EE Web Application Server, define a special annotation for this annotation type and configure the **uri** parameter value to **<none>**.

NOTE: Refer to WebLogic documentation for more information on supported security policies.

Inbound Web Service Deployment

An Inbound Web Service must be deployed to the J2EE Web Application Server in order for it to be available to the Web Service Clients to access the system. Refer to [Deploying Web Services](#) for more information.

Deploying XAI Inbound Service via IWS

For implementations using XAI inbound services for external messages, the product recommends moving to the inbound web service mechanism, which uses the J2EE Web Application Server to communicate with the product rather than the XAI servlet.

For XAI inbound services that use the **Business Adapter**, it is straight forward to move to IWS because the configuration is similar. In both cases, the service is configured to reference a business object, business service or service script. The associated WSDL for each record is similar. Changing the interface for the incoming message to use IWS instead of XAI inbound services is similar.

However, for XAI inbound services that use the **Core Adapter**, these services reference an underlying “page service” in the product. For these services, the Request and Response schemas for the XAI inbound service were created using the Schema Editor. In order to support calling an underlying “page service” in IWS, first a [business service](#) must be created to reference the page service (if one doesn’t already exist). However, the resulting schema for the business service is different from the Request and Response schemas related to the XAI inbound service. Moving this functionality to IWS using business services requires changes to the format of the incoming messages.

Moving all incoming messages over to use IWS instead of XAI is the product recommendation. However, to aid in implementations that have many integrations in place using the XAI inbound services that use the **Core Adapter** (or any

adapter whose message class is **BASEADA**), the product provides the ability to deploy these types of XAI inbound services to the J2EE Web Application Server along with the Inbound Web Services.

To take advantage of this capability, you must define a feature configuration option. Under the **External Messages** feature configuration type, the **Support XAI Services via IWS** is used to indicate if this feature is supported. Setting the value to **true** turns on the feature. If no option is defined for that option type, it is equivalent to setting the value to **false**.

When the system is configured to support XAI services via IWS, the [Inbound Web Service deployment](#) includes XAI inbound services (that are configured with an Adapter that references the **BASEADA** message class). The deployment portal will also include a zone showing the deployment status of these XAI Inbound Services.

Configuring Inbound Web Service Options

This topic describes the configuration needed for using inbound web services.

Technical Configuration

In order to use inbound web services, there are tasks a system administrator must perform.

Refer to the Server Administration Guide for technical details of each of these processes.

Maintaining Web Service Annotation Types

The product provides the annotation type **Policy Annotation** that defines the WS-Policy annotation. If your implementation wishes to define additional annotation types, use the Web Service Annotation Type portal. Open this page using **Admin > External Message > Web Service Annotation Type**.

You are taken to the query portal where you can search for an existing web service annotation type. Once an annotation type is selected, you are brought to the maintenance portal to view and maintain the selected record.

NOTE: Use of custom policies should only be considered if the policies supplied by the J2EE Web Application Server are not sufficient for your implementation's needs.

The **Web Service Annotation Type** zone provides basic information about the web service annotation type.

Please see the zone's help text for information about this zone's fields.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI_IWS_ANN_TYPE](#).

Maintaining Web Service Annotations

If your implementation wishes to define annotations, use the Web Service Annotation portal. Open this page using **Admin > Integration > Web Service Annotation**.

This is a standard [All-in-One portal](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI_IWS_ANN](#).

Maintaining Inbound Web Services

[Inbound Web Services](#) are used to define a specific message that your implementation will receive from an external system and provides configuration needed to process the inbound message.

The product provides several inbound web services out of the box. By default, no annotations are defined for the base inbound web services. You may modify the message options or the annotations for any base IWS record. In addition, you may define additional IWS records for other incoming messages supported by your implementation.

To view an inbound web service, navigate using **Admin > Integration > Inbound Web Service**. You are brought to a query portal with options for searching for inbound web services.

Once an inbound web service has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The Inbound Web Service zone displays the configuration information for the record, including its annotations, if applicable and its operations.

Deploying Web Services

This topic describes the configuration needed for using inbound web services.

Once an Inbound Web Service is defined it is not automatically available to the Web Service Clients to access the system. The Deployment Status and the Active flag (set to true) indicate whether a Web Service is available or not. The last step is to deploy the Inbound Web Services to the J2EE Web Application Server. This deployment phase has a number of steps that are automatically performed when a deployment is initiated:

- The Web Service files are generated and policies are attached.
- The WSDL is generated with appropriate annotations and enumerations.
- The necessary Java stub code to implement the Web Service in the J2EE Web Application Server is generated and compiled.
- The Web Services are built into a valid Web Application Archive (WAR) file.
- Optionally, the newly created Web Services WAR file is deployed to the J2EE Web Application Server. This can also be done manually for clustered deployments, if desired.

There are two methods available for deploying inbound web services:

- Deployment at the command line using the **ivsdeploy[.sh]** command as outlined in the Server Administration Guide. This method is recommended for native installations and production implementations.
- Deployment using the Inbound Web Service Deployment portal. This method is only supported in development (non-production) environments.

Inbound Web Service Deployment Portal

To use the online Inbound Web Service Deployment portal, navigate using **Admin > Integration > Inbound Web Service Deployment**.

The following sections describe the base zones that are provided on the portal.

Deploy Inbound Web Services

The Deploy Inbound Web Services zone provides information about the last deployment. If the region is a development (non-production) region you may use the **Deploy** button to deploy or re-deploy inbound web services. All inbound web services whose Active switch is Yes will be deployed. All whose active switch is No will be undeployed.

NOTE: When an Inbound Web Service is deployed, the value of its service revision field is captured. Certain changes to configuration will require re-deployment to take effect. When any of the following changes occur, the IWS service revision value is incremented. This will cause the deployment status to show **Needs Deploy**.

- Active switch is changed
- An Annotation is added or removed
- An Operation is added or removed.

- The Operation Name, Schema Type / Schema Name, Request or Response Schema, Request or Response XSL for an Operation is changed.

NOTE: In addition, if the implementation supports XAI services deployed through IWS, the appropriate XAI inbound services will also be deployed or undeployed as required.

Deployment Status

The Deployment Status zone displays a list of inbound web services in the product, including the deployment status.

The deployment status is determined by comparing the internal Service Revision field on each IWS against the value captured at the time of deployment.

- **Deployed.** Indicates that the IWS has been deployed and no changes have been detected to the configuration.
- **Needs Deploy.** Indicates that the IWS has never been deployed or has been deployed but in the meantime, changes have been detected to the configuration that require redeployment.
- **Undeployed.** Indicates that the IWS is marked as inactive and the IWS is not found to be deployed at this time.
- **Needs Undeploy.** Indicates that the IWS is marked as inactive but the IWS is found to be deployed at this time.

If the IWS has been deployed, the View column will include a **WSDL** link allowing you to launch a separate window to view the WSDL definition.

Use the broadcast button adjacent to any of the inbound web services listed in the zone to view the details of the IWS record. This causes the **Inbound Web Service** zone to appear. It is the same zone that appears on the [Inbound Web Service](#) maintenance portal.

XAI Inbound Service Deployment Status

The XAI Inbound Service Deployment Status zone is only visible if the feature configuration option **Support XAI Services via IWS** is configured on the **External Messages** feature type or if the system detects that there are XAI inbound services that have been deployed. (The latter condition is checked for the case where an implementation has XAI inbound services deployed and then chooses to discontinue using this functionality. After changing the feature configuration option to false, one more deployment is required to “undeploy” the XAI services.) The zone displays a list of XAI inbound services in the product that are related to page services. Refer to [Deploying XAI Inbound Service via IWS](#) for more information.

The deployment status is determined by comparing the record’s Version field against the value captured at the time of deployment.

- **Deployed.** Indicates that the XAI inbound service has been deployed and no changes have been detected to the configuration.
- **Needs Deploy.** Indicates that the XAI inbound service has not been deployed or has been deployed but in the meantime, changes have been detected to the configuration.
- **Undeployed.** Indicates that the XAI inbound service is marked as inactive or the **Support XAI Services via IWS** is not set to **true** and the XAI inbound service is not found to be deployed at this time.
- **Needs Undeploy.** Indicates that the XAI inbound service is marked as inactive or the **Support XAI Services via IWS** is not set to **true** but the XAI inbound service is found to be deployed at this time.

XAI inbound service does not have the equivalent of a Service Revision field that inbound web service has, which is only incremented when changes are made to the record that impact deployment. For XAI inbound service, the version number on the record is used. This field is incremented when any changes are made, even ones that may not impact deployment. As a result, some XAI Inbound Services may indicate “Needs Deploy” in cases where a redeployment may not be necessary. The recommendation when this occurs is to simply Deploy again to be safe.

If the IWS has been deployed, the View column will include a **WSDL** link allowing you to launch a separate window to view the WSDL definition.

Guaranteed Delivery

There are alternatives for sending messages to the system besides using inbound web services. An external system may be able to send messages to the system in a generic manner where a new web service does not need to be defined for every new type of message. These types of messages may provide a payload (the message) and the service script or business service to invoke. An example of this type of communication is a message sent from a mobile application using RESTful operations.

The external system may have no mechanism for retrying failed messages. For this situation, the product provides an algorithm that may be used to capture incoming messages that should 'guarantee delivery'. A servlet processing this type of message may invoke the [installation algorithm](#) - Guaranteed Delivery, passing the details of the message and an indication if a response should be returned. The algorithm is responsible for storing the message information in a table so that it can be subsequently processed.

NOTE: The framework does not provide a sample algorithm for this plug-in spot. Your specific product may provide an algorithm and additional support for guaranteeing messages. Refer to your product documentation for more information.

Outgoing Messages

"Outgoing messages" is the term used to describe messages that are initiated by our system and sent to an external system. Messages may be sent real time or near real time. The system provides the following mechanisms for communicating messages to external systems.

- **Outbound Messages.** This method allows implementers to use configurable business objects to define the message format and to use scripts to build the message. If sent near real-time the message is posted to the outbound message table waiting for Oracle Service Bus to poll the records, apply the XSL and route the message. If sent real time, the message dispatcher routes the message immediately.
- **Web Service Adapters.** Using a web service adapter, an implementation can consume a WSDL from an external system and create an "adapter" record that references the URL of the external system and creates appropriate request and response data areas to expose the payload information in a format understood by configuration tools. A script may then be written to build the request information and initiate a real-time web service call from within the system.
- **Send Email.** The system supplies a dedicated business service that may be used to send an email real-time from within the application.

All these methods are described in more detail in the following sections.

Outbound Messages

Outbound messages provide functionality for routing XML messages to an external system real-time or in near real time. In addition the functionality supports collecting related messages into a batch to then be sent to an external system as a consolidate XML message.

For each outbound message that your implementation must initiate you define a [business object](#) for the outbound message maintenance object. Using the business object's schema definition, you define the fields that make up the XML source field. These are the fields that make up the basis for the XML message (prior to any XSL transformation).

Each outbound message requires the definition of its schema by creating a business object whose schema describes the information that is provided to the external system. An XSL transformation may then be performed when routing the message to an external system.

For each external system that may receive this message, you configure the appropriate message XSL and routing information.

Because the outbound message type is associated with a business object, your implementation can easily create outbound message records from a script using the [Invoke business object](#) step type. Such a script would

- Determine the appropriate [outbound message type](#) and [external system](#) based on business rules
- Access the data needed to populate the message detail
- Populate the fields in the schema and use the **Invoke business object** script step type for the outbound message type's business object to store the outbound message.
- The resulting outbound message ID is returned to the script and the scriptwriter may choose to design a subsequent step to store that ID as an audit on the record that initiated this message.

The following topics provide more information about functionality supported by outbound messages.

NOTE: For implementations that continue to use MPL and XAI, there is additional functionality related to outbound messages. Refer to [Outgoing Messages](#) for more information.

Polling Outbound Messages Using OSB

If the outbound message that needs to be sent to an external system can be sent as an asynchronous message (in 'near real time'), the process initiating the outbound message should create a record in the outbound message staging table. Oracle Service Bus (OSB), is the recommended tool to use to process outbound messages in near real-time.

Outbound messages that should be processed by OSB should be configured with a processing method defined as **SOA** for the external system / outbound message type. No other information is required for defining outbound message types that are processed by OSB.

For the OSB part of the processing, the product provides a custom transport: OUAUF Outbound Message that may be used by an implementation to define messages to process and how to process them.

This section provides an overview of steps required to develop OSB integrations for outbound messages created by your product.

Before developing OSB integrations, a developer should be familiar with OSB development such as creating proxy services, business services, and message flow/routing. These terms are defined as follows:

Proxy Service: In OSB, a Proxy Service is the entity that processes a given type of message and routes it to a Business Service. A separate proxy service should be defined for each type of outbound message. If a given outbound message type may be routed to different external systems, it is the responsibility of the proxy service to query the external system defined on the outbound message and invoke the appropriate business service (see below). If any transformation is required prior to routing a message to a business service, it is the proxy service's responsibility to perform the transformation.

Business Service: In OSB, a Business Service is an entity that receives a message from OSB and routes it to the appropriate destination. This should not be confused with the Business Service object provided in the product in the configuration tools.

FASTPATH: Refer to the whitepaper *OSB Integration* for more information.

Batch Message Processing

Your implementation may be required to send messages to the same destination as a single XML file with multiple messages include. The following points describe this logic:

- The individual messages that should be grouped together must have a processing method of **batch** on the external system / outbound message type record. The appropriate batch code that is responsible for grouping the messages must also be provided.
- A separate "consolidated message" outbound message type should be configured for the external system with a processing method of **SOA**.
- When outbound message records are created for the individual messages, the batch code and current batch run number are stamped on the record.

- When the batch process runs it is responsible for building the XML file that is a collection of the individual messages. This batch process should include the following steps:
 - Format appropriate header information for the collection of messages
 - Apply the individual message XSL to each message before including the message
 - Insert a new outbound message for the external system with the "consolidated message" outbound message type.
- The consolidated message is ready to be processed by Oracle Service Bus.

NOTE: No process provided. The system does not supply any sample batch job that does the above logic.

Real Time Messages

The system supports the ability to make web service calls, i.e. sending real time messages, to an external system.

The system supports special functionality for sending an Email message real-time. Refer to [Sending Email](#) for more information.

For other types of real-time messages, the system also uses outbound message type and external system configuration to format and route the message. When defining the configuration for real time messages, an additional step is required to define the mechanism for routing the message using a message sender. The system supports routing messages via HTTP and via JMS. Note that for HTTP routing, the system also supports sending the message using a JSON format.

Just like near real-time messages, initiating a real-time outbound message may also be done from a script. When a real time message is added, the system immediately routes it to the external system. If the external system provided a response message back, the system captures the response on the outbound message. If the outbound message type for the external system is associated with a response XSL it is applied to transform the response. In this case the system captures the raw response as well on the outbound message. Note that the outbound message BO should be configured to capture a response XML in its schema.

Any error (that can be trapped) causes the outbound message to be in a state of **Error**. It is the responsibility of the calling process to check upon the state of the outbound message and take a programmatic action. When the outbound message state is changed back to **Pending** the message will be retried.

The base package provides two business services: Outbound Message Dispatcher (**F1-OutmsgDispatcher**) and Outbound Message Mediator (**F1-OutmsgMediator**) that further facilitate making web service calls. Both business services are similar, allowing the calling script to configure the following behavior (with differences noted):

- Whether or not exceptions encountered while sending the message are trapped. Trapping errors allows the calling script to interrogate any errors encountered and take some other programmatic action.
- Whether or not the sent message is persisted as an actual outbound message record. If the message should not be persisted, the two business services handle this differently. The Outbound Message Dispatcher temporarily creates an outbound message record and subsequently deletes it. Because an outbound message is created, all the business object algorithms are executed. However, there is a performance implication. In contrast the Outbound Message Mediator sends a non-persisted message without creating and deleting an outbound message record. It is more efficient, but should only be used if no algorithms on the outbound message BO are needed.

Refer to the descriptions of the two business services for more information.

Designing the System for Outbound Messages

The following sections describe the setup required when using [outbound messages](#) to communicate with an external system. The configuration walks you through the steps to configure a single external system and all its messages.

Define the Outbound Message Type

For each outbound message that must be sent to an external system, create a [business object](#) for the outbound message maintenance object. Using the business object's schema definition, your implementation defines the elements that make up the XML Source field (XML_SOURCE). These are the elements that are the basis for the XML message. XSL transformations may be applied to this XML source to produce the XML message.

If your integration is real-time and a response is expected, the Outbound Message business object should also map to the XML Response field (XML_RESPONSE).

- You may decide to capture the response as is and define the element as “raw”. For example

```
<responseDetail mapXML="XML_RESPONSE" type="raw"/>
```

In this scenario, a Response XSL may or may not be needed.

- Alternatively, if the details of the response are needed, you may define specific elements for the response. For this option, depending on how the integration is designed, a Response XSL may be needed to transform the response into the expected XML format.

Once you have your business object and schema, define an [outbound message type](#) for each unique outbound message.

Real-Time Message Configuration

When messages are routed to an external system real-time using the outbound message dispatcher or using the real-time send email business service. The system supports routing messages using HTTP and routing messages using JMS. In addition, there is a special type of message sender used for sending emails. The following sections highlights the supported real-time communication and the configuration needed for each.

Email Messages

For sending emails, the following configuration is needed:

- Define a [message sender](#) configured for email. Senders of this type should be configured with the **RTEMAILSND** class. The sender context is used to configure the connection information for the connecting to the SMTP server.
- This sender may be defined as the default email sender on the [message option](#) table. Alternatively, the message sender can be provided to the business service as input. Refer to [Sending Email](#) for more information.

Outbound Messages

For other outbound messages that are routed using the real-time outbound message dispatcher, a [message sender](#) must be configured to define how to route the message. The following points highlight more detail related to this configuration.

Determine the communication mechanism prior to configuring the sender.

- When routing the message using JMS, the following configuration must be defined
 - Define an appropriate [JNDI Server](#) that indicates where the JMS resources are located.
 - Define a [JMS Connection](#) to define additional configuration needed for the connection.
 - Define the [JMS Queue](#) or [JMS Topic](#) to define the queue or topic to use.
- When communicating using a JSON format, determine the method to use to convert the XML to JSON. The desired method is driven by how the request should be sent.
 - If choosing the **Base JSON Conversion** method, if XSL transformation needs to be applied prior to the conversion to JSON, then the target XML Request Schema must be defined (using a data area) so that the conversion logic knows the format of the XML it is converting. The XSL is applied to the outbound message's XML Source, resulting in the defined XML Request Schema, which is then converted to JSON. If XSL transformation is not required, then the outbound message's XML Source is converted to JSON.

- If the XML source on the outbound message can be converted to JSON using an XSL, then the **XSL Transformation** method may be chosen.
- You may also choose to convert the XML Source to JSON via the **Standard API Conversion** method (using a Jettison library). With this method, an XSL may optionally be provided. The conversion will be performed on the transformed XML.
- For the response, if the outbound message BO defines detailed elements for the XML Response field, then the JSON should be converted to this format. If your conversion method is **Base JSON Conversion**, then if the JSON response cannot be converted directly to the XML Response elements on the outbound message BO, then define a Response Schema (data area) that represents the results for the base JSON conversion. In addition, define an XSL that can transform the response from the converted XML to the XML format expected on the BO. If the conversion method is **Standard API Conversion** or **XSL Transformation**, the standard API is used to convert JSON to XML. An XSL may be defined to convert the response to the XML Response if needed.
- If the outbound message BO defines a “raw” element to capture the response, then a response schema and XSL are not necessary. In this case, the system will perform a JSON to XML conversion using the **Standard API Conversion** method (regardless of the conversion method defined) and the result is captured in the XML response.
- When using HTTP, no additional configuration is needed prior to configuring the sender.

Define a [message sender](#) configured for each appropriate routing method. The invocation type should be configured as **Real-time**. For routing via HTTP, use the **RTHTTPSNDR** — HTTP sender class. For routing via HTTP using JSON format, use the **RTJSONSNDR** — JSON sender class. For routing via JMS, use the **RTJMSQSNDR** — JMS sender class and configure the JMS Connection and JMS Queue or JMS Topic. Use the sender context to configure the required values for connecting to the appropriate destination.

Configure the [external system](#) / outbound message type collection. The processing method defined for the external system / outbound message type must be **Real-time**.

Define the External System and Configure the Messages

Define an [external system](#) and configure the valid outgoing messages and their method of communication (processing method). Refer to [Outbound Messages](#) for more information.

Outbound Message Schema Validation

The outbound messages that are generated by the system should be well formed and valid so that they do not cause any issues in the external system. To ensure this validity you may configure the system to validate messages before they are routed to their destination. Note that the validation is applied just before communication with the sender and therefore after any Request XSL has been applied.

- Define a directory where the valid W3C schemas are located using the Message option **Outbound Message Schema Location**
- Each [external system](#) message must indicate the appropriate W3C schema to validate against

You may turn on or off this validation checking using the Message option **Schema Validation Flag**.

Configuring the System for Outbound Messages

The following sections describe the setup required when using [outbound messages](#) to communicate with an external system.

JNDI Server

If using JMS to communicate outbound messages, define a new JNDI Server. Open **Admin > Integration > JNDI Server**.

Description of Page

Enter a unique **JNDI Server** and **Description**.

Indicate the Provider URL to indicate the location of the JNDI server.

Indicate the **Initial Context Factory**, which is a Java class name used by the JNDI server provider to create JNDI context objects.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_JNDI_SVR](#).

JMS Connection

To define a JMS Connection, open **Admin > Integration > JMS Connection**.

Description of Page

Enter a unique **JMS Connection** and **Description**.

Indicate the **JNDI Server** to be used. Refer to [JNDI Server](#) for more information.

Use the **JNDI Connection Factory** to indicate the lookup keyword in the JNDI server used to locate the JMS connection.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_JMS_CON](#).

JMS Queue

To define your JMS Queue values, open **Admin > Integration > JMS Queue**.

Description of Page

Enter a unique **JMS Queue** and **Description**.

Enter the **Queue Name** as defined in the JNDI server. This is the JNDI lookup name identifying the queue.

Use the **Target Client Flag** to indicate whether or not the target client is **JMS** or **MQ**.

Select the **JNDI Server** where the queue is defined. Refer to [JNDI Server](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_JMS_Q](#).

JMS Topic

To define your JMS Topic values, open **Admin > Integration > JMS Topic**.

Description of Page

Enter a unique **JMS Topic** and **Description**.

Select the **JNDI Server** where the topic is defined. Refer to [JNDI Server](#) for more information.

Enter the **Topic Name** as defined in the JNDI server. This is the JNDI lookup name identifying the topic.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_JMS_TPC](#).

Message Sender

The topics in this section describe the maintenance of a message sender

Message Sender - Main

To define a new sender, open **Admin > Integration > Message Sender**.

Description of Page

Enter a unique **Message Sender** and **Description**.

Set **Invocation Type** to **Real-time**.

NOTE: The invocation type **MPL** remains in the product for upgrade purposes but is not recommended.

Indicate the **Message Class** for this sender, which indicates the method used to route the message. The real-time sender classes are **RTEMAILSNDR** — Email sender, **RTHTTPSNDR** — HTTP sender, **RTJMSQSNDR** — JMS sender and **RTJSONSNDR** — HTTP JSON sender.

The following sender classes are related to MPL processing and remain in the product for upgrade purposes: **DWNSTGSNDR**- Download Staging sender, **EMAILSENDER**- Email sender, **FLATFILESNDR**- Flat file sender, **HTTPSNDR**- HTTP sender, **JMSENDER**- JMS Queue sender, **OUTMSGSNDR**- Outbound Message sender, **STGSENDER**- Staging Upload sender, **TPCSNDR**- JMS Topic sender and **UPLDERRHNDLR**- Upload Error Handler.

Indicate whether or not this sender is currently **Active**.

Indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**.

If the Message Class is one that connects to a JMS Queue or JMS Topic, indicate the appropriate **JMS Connection**

FASTPATH: Refer to [JMS Connection](#) for more information.

If the Message Class is one that connects to a JMS queue, indicate the name of the **JMS Queue** to define where the message is to be sent.

FASTPATH: Refer to [JMS Queue](#) for more information.

If the Message Class is one that connects to a JMS topic, indicate the name of the **JMS Topic** to define where the message is to be sent.

FASTPATH: Refer to [JMS Topic](#) for more information.

The **XAI JDBC Connection** remains for legacy purposes but is not applicable for supported functionality.

Message Sender - Context

The sender may require context information to define additional information needed by the system to successfully send outgoing messages. Open **Admin > Integration > Message Sender** and navigate to the **Context** page.

Description of Page

Define the **Context Type** and **Context Value**, which contain parameters for senders when more information is required. See below for the supported context values for different types of senders.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_SENDER](#).

Email Sender Context

The email sender is used by the business service that [sends email messages real-time](#).

An email sender must point to the Message Class **RTHTTPSNDR**. In addition, the following context records should be defined for senders of this type.

Context Type	Description
SMTP Host name	The SMTP server host name.
SMTP Username	The user ID used to access the SMTP server.

Context Type	Description
SMTP Password	The password used to access the SMTP server.
Response Time Out	The amount of time the system should wait for a real time response.

HTTP Sender

An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. HTTP senders should reference a Message Class of **RTHTTPSNDR** or **RTJSONSNDR**.

Various parameters are required to establish a session with the target HTTP server. You specify these parameters by defining a collection of context values for the sender. A set of context types related to HTTP variables is provided with the product. The following section describes the context types and where appropriate, indicates valid values.

Before defining the HTTP sender, you need to find out how the HTTP server on the other side expects to receive the request, and in particular, to answer the following questions:

- What is the address of the HTTP server?
- Is the HTTP server using a POST or GET HTTP method?
- If the server is using POST, how are message contents passed? Does it use an HTTP FORM or does it pass the data in the body of an XML message?

Context Type	Description	Values
HTTP URL1 - URL9	Used to construct the URL of the target HTTP server. Since the URL may be long and complex, you can break it into smaller parts, each defined by a separate context record. The full URL is built by concatenating the values in URL1 through URL9. You may use substitution variables when entering values for URL parts. Note that the substitution string @XMLMSG@ may be used for GET calls if an XSL has been applied to convert the message into HTTP GET parameters. It is useful if the HTTP Form is not applicable to the type of message.	
HTTP Method	The HTTP method used to send the message.	POST or GET
HTTP Proxy Host	If connecting to the remote system requires using an HTTP Proxy, then this context field can be used to configure the HTTP Proxy Host. If the Proxy Host is set, the Sender class must use the value specified to connect to the remote system via a proxy.	
HTTP Proxy Port	If connecting to the remote system requires using an HTTP Proxy, then this context field can be used to configure the HTTP Proxy Port. If the Proxy Port is set, the Sender class must use the value specified to connect to the remote system via a proxy. If the HTTP Proxy Host is not set, HTTP Proxy Port is ignored	

Context Type	Description	Values
HTTP Transport Method	<p>and the connection will be made directly to the remote system.</p> <p>Specifies the type of the message. You can either send the message or send and wait for a response.</p>	Send or sendReceive
HTTP Form Data	<p>Used when the message is in the format of an HTML Form (<code>Content-Type: application/x-www-form-urlencoded</code>).</p> <p>This context specifies the form parameters (data) that should be passed in the HTTP message. Since a form may have multiple parameters, you should add a context record for each form parameter.</p> <p>The value of a form parameter takes the format of <code>x=y</code> where <code>x</code> is the form parameter name and <code>y</code> is its value.</p> <p>If <code>y</code> contains the string <code>@XMLMSG@</code> (case sensitive) then this string is replaced by the content of the service response XML message. The <code>@XMLMSG@</code> string can be used in the HTTP Form Data or in the HTTP URL, but not in both.</p> <p>If a context record of this type is defined for a sender, the sender uses the HTML Form message format to send the message even if <code>@XMLMSG@</code> is not specified in one of the context records.</p> <p>If a context record of this type is not defined for a sender, then the XML is sent with <code>Content-Type: text/plain</code>. When using POST it is put in the HTTP message body.</p> <p>Always required when using the GET method. If you are using the GET method and do not specify a Form Data context record, no message is transferred to the HTTP server.</p> <p>The MPL server builds <code>formData</code> by concatenating the individual parts.</p> <p>You may use substitution variables when entering values for Form Data.</p>	
HTTP Login User	<p>The HTTP server may require authentication. Add a context record of this type to specify the login user to use.</p>	
HTTP Login Password	<p>The HTTP server may require authentication. Add a context record of this type to specify the login password to use.</p>	
HTTP Header	<p>Sometimes the HTTP server on the other side may require the addition of HTTP headers to the message.</p>	

Context Type	Description	Values
	For each HTTP header that has to be specified you should add a context record with a value having the following format x:y where x is the header name and y is the value for the header	
HTTP Time Out	Indicates the amount of time to wait for a connection to be established with the remote system.	
Character Encoding	Indicates if the message should be encoded. The sender will add to the HTTP's content type header the string ; charset=x where x is the value of this context and when sending the message it will encode the data in that encoding.	UTF-8 or UTF-16
Response Time Out	The amount of time the system should wait for the remote system to send a response.	

JMS Sender

A JMS sender is one that sends messages to a JMS queue or JMS topic. JMS senders should reference a Message Class of **RTJMSQSNDR**.

The following parameters are used to connect to the JMS resource.

Context Type	Description	Values
JMS Message Type (Bytes(Y)/Text(N))	Indicates whether the data is sent as a bytes message or as a text message.	Y or N
JMS User Name	Enter the user name to connect to the JMS resource.	
JMS User Password	Enter the password to use to connect to the JMS resource.	
JMS Header	If JMS header values are required for the message, use this context type. For each JMS header that has to be specified, add a context record with a value having the following format x:y where x is the header name and y is the value.	

Defining Outbound Message Types

Use this page to define basic information about an outbound message type. Open this page using **Admin > Integration > Outbound Message Type**.

NOTE: This page is not available if the **External Message** module is [turned off](#).

Description of Page

Enter a unique **Outbound Message Type** and **Description**. Use the **Detailed Description** to describe the outbound message type in detail.

Indicate the **Business Object** that defines business rules and the schema for outbound messages of this type.

Indicate the relative **Priority** for processing outbound message records of this type with respect to other types.

This bottom of this page contains a [tree](#) that shows the various objects linked to the outbound message type. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI_OUTMSG_TYPE](#).

External Systems

Use this page to define an external system and define the configuration for communication between your system and the external system.

External System - Main

Open this page using **Admin > Integration > External System**.

NOTE: This page is not available if both the **External Message** and the **Open Market Interchange** modules are [turned off](#).

Description of Page

Enter a unique **External System** and **Description**.

Use the field **Our Name In Their System** to specify the identity of your organization (i.e., your external system identifier) in the external system.

NOTE: The workflow process profile and notification download profile are only applicable to products that support workflow and notification. They are not visible in the product if the **Open Market Interchange** module is [turned off](#).

If this external system sends inbound communications through notification upload staging, the type of workflow process that is created is controlled by the sender's **W/F (Workflow) Process Profile**.

If you send notifications to this external system, select a **Notification DL (download) Profile** that is used to define the configuration of the outgoing messages.

NOTE: The remaining fields are not visible if the **External Message** module is [turned off](#).

Set **Usage** to **Template External System** for external systems whose outbound message type configuration is inherited by other external systems.

If the outbound message type configuration should be inherited from a template external system, define the **Template External System**. If this field is specified, the outbound message type collection displays the data defined for the template system as display-only.

The **Outbound Message Type**[accordion](#) contains an entry for every type of outbound message defined for this external system. For each type of outbound message identify its **Outbound Message Type**.

Define the **Processing Method** for messages of this type. Valid values are **Batch**, **Real-time**, **SOA** and **XAI**.

Define an appropriate **Message Sender** if the processing method is **XAI** or **Real-time**.

Define an appropriate **Batch Control** if the processing method is **Batch**.

If the message sender is one with a message class of **RTJSONSNDR**, indicate the **JSON Conversion Method**. The valid values are **Base JSON Conversion**, **Standard API Conversion** and **XSL Transformation**. Refer to [Real Time Message Configuration](#) for more information about these methods and the additional configuration that may be applicable.

If the **JSON Conversion Method** is **Base JSON Conversion**, the **Request Schema** is enabled. Populate a data area that defines the schema for the XML format to convert the outbound message's BO schema to prior to performing the JSON conversion. Refer to [Real Time Message Configuration](#) for more information.

The **Message XSL** is the schema used to transform information from the format produced by the system to a format understood by the sender, who receives a message of this type. This is not applicable for Processing Method of **SOA**.

Enter the file name of the appropriate **W3C Schema** if you want to validate the message built for outbound messages for this external system / outbound message type prior to being routed to their destination. Refer to [Outbound Message Schema Validation](#) for more information. This is not applicable for Processing Method of **SOA**.

If the **JSON Conversion Method** is **Base JSON Conversion**, the **Response Schema** is enabled. Populate a data area that defines the schema for the XML format that the JSON message is initially converted to. The XML is then converted to BO XML. Refer to [Real Time Message Configuration](#) for more information.

Response XSL will have the same search service as is used for the existing Message XSL field. This field will only be displayed when the processing method is **Real-time**. Refer to [Real Time Messages](#) for more information on how it is used.

External System - Template Use

If you are viewing an external system whose usage is a **Template External System**, use this page to view the other external systems that reference this one. Open this page using **Admin > Integration > External System** and then navigate to the **Template Use** tab.

Description of Page

The tree shows every external system that references this external system as its template.

Message Option

The Message Option page defines various system settings used by the system when processing external messages.

Note that some of the options are only applicable to implementations still using the XAI and MPL servers. The settings here may be overridden by the [AdHoc Parameters](#) section of the XAIParameterInfo.xml or MPLParameterInfo.xml.

To define options for your environment, open **Admin > Integration > Message Option**.

Description of Page

The following options are supported.

Option	Description	MPL / XAI Option Name
Automatically Attempt Resend to Unavailable Sender (Y/N)	Set to Y if you wish to enable Automatic Resend . Set to N if you wish to log errors when the system fails to send an outgoing message.	shouldAutoResend
Default Email Sender	This is the default Message Sender used for sending e-mails when no explicit Message Sender is specified.	defaultEmailSender
Default Response Character Encoding	Determines the character encoding to be used when a response is sent. For example, you may specify UTF-8 or UTF-16 . If no value is specified then the default is UTF-8 . If no special encoding should be done, then enter the value none .	defaultResponseEncoding
Default User	The default user is used by XAI to access your product when no other user is explicitly specified. Refer to Server Security for more information. Additionally, the Default User is used for MPL transactions where there is no facility to provide a User ID. For example, no facility exists to provide a user id when reading messages from a JMS Queue. In these messaging scenarios, the system will use the Default User for authorization purposes.	defaultUser

Email Attachment File Location	This is the default location of e-mail attachment files. If not specified, the e-mail service provided with the product assumes a full path is provided with each attachment file.	emailAttachmentFileLocation
Email XSL File Location	This is the default location of e-mail XSL files. If not specified, the e-mail service provided with the product assumes a full path is provided to an XSL file as part of an e-mail request.	emailXSLFileLocation
JDBC Connection Pool Max size	The MPL uses a pool of JDBC connections to connect to the database. This option determines the maximum number of JDBC connections that can be opened in the pool. The default value is 100.	JDBCConnPoolMaxSize
Maximum Errors for a Sender	This value is required if you have enabled Automatic Resend . It defines how many errors you receive from a sender when attempting to send an outgoing message before you mark the sender unavailable .	maxSendingErrors
Messages JDBC Connection	Specifies the JDBC connection that XAI uses to read the text for its messages.	messagesJDBCCConnection
Messages Language	The default language to use for the messages.	language
MPL Administrator Port	The port number to be used for receiving MPL operator commands.	adminPort
MPL HTTP Server Authentication Method	This setting, along with the MPL HTTP Server User and Password are used to secure commands received by your MPL (such as those issued via XAI Command) through HTTP. Currently only BASIC authentication is supported.	MPLHTTPAuthMethod
MPL HTTP Server Password	This setting, along with the MPL HTTP Server Authentication Method and User are used to secure commands received by your MPL (such as those issued via XAI Command) through HTTP. The password should be in encrypted form, using the same encryption that is used for the database password. .	MPLHTTPAuthPassword
MPL HTTP Server User	This setting, along with the MPL HTTP Server Authentication Method and Password are used to secure commands received by your MPL (such as those issued via XAI Command) through HTTP.	MPLHTTPAuthUser
MPL Log File	The MPL Log File setting is used to specify the name of the file where MPL log information is to be written. The log contains technical information about the operation of the MPL.	MPLLogFile
MPL Trace File	The MPL Trace File setting is used to specify the name of the file where MPL trace information is to be written.	MPLTraceFile
MPL Trace Type	The MPL Trace Type is used to enable or disable tracing of the MPL. The possible values are FULL - All trace messages are written to the log file and NOLOG - No information is written to the log file.	MPLTraceType
Outbound Message Schema Location	Enter the full path of the virtual directory where valid W3C schemas are stored if your implementation wants to validate outbound message schemas . For example: http://localhost/cisxai/schemas.	xaiOuboundSchemaLoc
Privileged Users	Comma separated list of users that are allowed to specify an effective User or effective User Id via framework custom SOAP Headers.	superUsers

Records MPL Receiver Will Process At a Time	If your implementation has configured multiple MPL servers , indicate the number of records that each MPL receiver should process.	Not currently used
Schema Directory	The full path of the virtual directory where XML schemas are stored. For example: http://localhost/cisxai/schemas. If this option is not specified, the XAI uses the current directory, from where it is being run, to locate schemas.	schemaDir
Schema Validation Flag	Enter Y to turn on schema validation for outbound messages . Enter N to turn this off.	xaiSchemaValidationCheck
Send SOAP Fault as HTTP 500	Enter Y to ensure that a SOAP error is reported as an HTTP 500 "internal server error".	sendErrorAsHttp500
Sender Retry Seconds	This value is required if you have enabled Automatic Resend . It defines how many seconds to wait after marking a sender unavailable before you mark the sender available again (and retry sending messages to it).	senderWaitTime
System Error JDBC Connection	When a request fails to execute due to a system error, the MPL retries its execution several times. The MPL registers the system error in a table and uses this table for the retries. This setting specifies the JDBC connection required to access this table. Only enter a value in this field if it is different from the database environment used to read the XAI registry.	systemErrorTableJDBCConnection
System Error Max Retry	When a request fails to execute due to a system error, the MPL retries its execution several times until a maximum number of retries is reached. This option specifies the maximum number of retries.	systemErrorMaxRetries
System Error Retry Interval	When a request fails to execute due to a system error, the MPL retries its execution several times. This option specifies the number of seconds the MPL server waits between retries.	systemErrorRetryInterval
Thread Pool Initial Size	The MPL uses a thread of pools to enhance performance. The MPL starts with a minimum number of threads and grows/shrinks the pool based on the MPL system load. This option specifies the initial number of threads in the thread pool. The minimum number of threads is 12.	threadPoolInitialSize
Thread Pool Max Size	This option specifies the maximum number of threads in the thread pool.	threadPoolMaxSize
Thread Pool Non Activity Time	This option specifies how long a thread in the pool may be inactive before it is timed out and released from the pool.	poolNoneActivityTime
To Do Type for Inbound JMS Message Errors	To Do type for inbound JMS message errors. The inbound message processor uses this To Do type when creating To Do entries for inbound JMS messages that cannot be successfully processed. The system provides the To Do type F1-INJMS that may be used here.	toDoTypeforInboundJMSMessageErrors
To Do Type for Outbound Message Errors	To Do type for outbound message errors. The outbound message receiver uses this To Do type when creating To Do entries for outbound messages that cannot be successfully processed. The system provides the To Do type F1-OUTMS that may be used here.	outboundErrorToDo

WSDL Service Address Location	Specifies the SOAP address location that XAI uses in creating a WSDL. If no value is present, the XAI's URL is used.	wsdlAddressLocation
XAI Authentication Password	The multi-purpose listener uses this field in combination with the XAI Authentication User when attempting to communicate with the XAI server over HTTP, which is running on a secured servlet and requires authentication.	HTTPBasicAuthPassword
XAI Authentication User	The multi-purpose listener uses this field in combination with the XAI Authentication Password when attempting to communicate with the XAI server over HTTP, which is running on a secured servlet and requires authentication.	HTTPBasicAuthUser
XAI Trace File	The full path name for the file, where the XML messages should be written. For example: c:\inetpub\wwwroot\cisxai\xai.log.	traceFile
XAI Trace Type	Use this option to specify the level of logging. The possible values are FULL - All XML messages are written to the log file and NOLOG - No information is written to the log file. FASTPATH: Refer to Server Trace for more information about tracing.	traceType
XSL Directory	The full path of the virtual directory where XSL transformation scripts are located. XSL transformation scripts can be defined for each service. By default, this is the same directory as the schemas directory.	XSLDir

Where Used

Used by the XAI tool to obtain various required settings and locations.

Managing Outbound Messages

Use this page to view information about outbound messages.

Outbound Message - Main

Open this page using **Menu > Integration > Outbound Message**.

Description of Page

Outbound Message ID is the system-assigned unique identifier of the outbound message. These values only appear after the outbound message is added to the database.

The **Processing Method** indicates whether this record will be processed by a **Batch** extract process, through the **XAI** tool or **Real-time**. The value defined on the external system / outbound message type collection populates this value.

When records are created with a processing method of **Batch**, the system sets Extract to **Can Be Extracted**. Change the value to **Not to be extracted** if there is some reason that processing should be held for this record.

For records with a processing method of **Batch**, **Batch Control** indicates the process that will extract this record. This value is populated based on the on the external system / outbound message type's value. **Batch Number** indicates in which batch run this record was extracted or will be extracted.

The **Retry Count** is used by the XAI tool to keep track of how many times the tool tried to process this record and could not process the record, resulting in an error.

The **Creation Date** indicates the date that this record was created.

If the processing method is **XAI**, **Status** defines the state of the outbound message record. Refer to [Lifecycle of Outbound Message](#) for more information.

For messages in **error** status, click **Pending** to change the status back to pending for reprocessing.

For messages in **pending**, **error**, or **in progress** status, click **Cancel** to cancel the message and prevent further processing.

Outbound Message - Message

Use this page to view the XML source used to build an outbound message. Open this page using **Menu > Integration > Outbound Message** and then navigate to the **Message** tab.

Description of Page

The **XML Source** is displayed.

If a message XSL is defined on the external system / outbound message type record linked to this outbound message, the **Show XML** button is enabled. Click this button to view the XML that is a result of applying the Message XSL to the XML source.

Outbound Message - Response

Use this page to display the XML response. Open this page using **Menu > Integration > Outbound Message** and then navigate to the **Response** tab.

Description of Page

The **XML Response** and optionally the **XML Raw Response** is displayed.

XML Response displays the response data from the system called by the real-time message. If a response XSL is defined on the external system / outbound message type record linked to this outbound message, a transform is performed and the XML Raw Response displays the original, unchanged response.

Web Service Adapters

The base product provides a configuration object called Web Service Adapter that is used to help build configuration objects to allow for functionality in the system to initiate a web service call from within the system. A Web Service Adapter provides the following functionality:

- WSDL (web service description language) import. An implementer can use the WSDL import functionality to read the details of a WSDL into the system
- Internal API generation. The system generates internal data areas that have two main purposes: they provide the API for custom code to define the appropriate input and they provide output data for the web service call using Oracle Utilities Application Framework schema language. In addition, the web service dispatcher uses element mapping defined in the data areas to transform the internal XML into the structure expected by the external system as described in the WSDL.
- Defines the URL needed to perform the web service call at runtime.

Understanding Web Service Adapters

The following topics describe the system functionality in more detail.

Importing a WSDL

Configuring a Web Service Adapter starts by identifying the WSDL (the web service description language document used to define the interface) that will be provided by the external system. The following steps describe the base product functionality provided to allow a user to import a WSDL.

- Navigate to the **Web Service Adapter** page in add mode and select the appropriate base business object.
- Enter a meaningful Web Service Name and appropriate descriptions.
- Provide the URL of the given WSDL.
- Click **Import** to retrieve the details of the WSDL. The system then parses the WSDL details and populates the WSDL Service Name, WSDL Source, WSDL Port, URL and a list of Operations (methods) defined in the WSDL.
- Determine which Operations should be **active** based on the business requirements for invoking this web service. **Active** operations are those that the implementation is planning to invoke from the system. These require appropriate request and response data areas generated for them. The following section provides more information about that.
- Specify the appropriate Security Type to configure the type of security to use when invoking this web service.
- Click Save.

At this point, a web service adapter record is created in pending status. The next step is to generate the request and response data areas for the operations configured as active.

Generating Request and Response Data Areas

Each **active** operation for the web service adapter requires a pair of data areas, request and response, that represent the request and response XML messages for the operation.

The base product provides steps to generate the data areas as follows:

- As described in the Importing a WSDL section above, the operations listed in the WSDL are generated for the web service adapter and the implementer should indicate which operation to activate.
- After saving the **pending** web service adapter, the display lists all the active operations and for each one includes a **Generate** button.
- After clicking **Generate** for an operation, a window appears where the names of the new Request and Response Data Areas may be defined. Click **Save** to generate the data areas.

The generated data areas provide the API for the implementer to use when implementing the web service call in an appropriate algorithm or service in the system. The data areas contain the appropriate mapping from the elements that the implementer works within the code that invokes the web services and the WSDL definitions.

To facilitate generating the request and response data areas, the base product invokes a special business service used to create the appropriate mapping. The business service is defined as a BO option on the Web Service Adapter business object. This allows an implementation to provide a custom business service to further enhance the request and response mapping where appropriate.

NOTE:

Generated data areas. It is possible to edit and modify the generated data areas after they are created. An implementer can change element names or remove unneeded elements if desired. Manually changing the generated data areas must be done only when absolutely necessary. This is because the system is not able to validate manual changes and issues with the data areas would only be detected at run time.

Activating Web Service Adapters

The business objects provided by the base package for web service adapters include a simple lifecycle of **Pending** and **Active**. Configure the web service adapter and its data areas while in **Pending** status and activate it when it is ready to be implemented in the appropriate system functionality.

Invoking Web Services

To make a call to a web service using a web service adapter, the system has provided a Web Service Dispatcher business service (**F1-InvokeWebService**) to submit a web service call. The calling program is responsible for retrieving all the information to correctly populate the request data required by the web service call before invoking the business service.

NOTE:

Refer to the detailed description of the business service for more information.

Limitations

The following points highlight limitations associated with the types of web services that the system supports:

- It is possible for one WSDL document to contain definitions for several web services. The system currently supports only one port or service per WSDL document.
- It is possible for a WSDL to support multiple message patterns. The system currently supports only request / response.

Setting Up Web Service Adapters

Use the Web Service Adapter portal to define the configuration needed to communicate with an external system using a web service call. Open this page using **Admin > Integration > Web Service Adapter**. You are brought to a query portal with options for searching for web service adapters.

Once a web service adapter has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Web Service Adapter** zone provides basic information about the web service annotation type.

Please see the zone's help text for information about this zone's fields.

FASTPATH: Refer to [Understanding Web Service Adapters](#) for information about common web service adapter functionality.

Sending Email

The framework provides the ability to initiate an email from within the system. The following topics highlight the functionality available.

- Sending email “real time” using a specific business service. The framework provides a business service **F1–EmailService** that supports sending an email. The schema supports elements for all the information required to create an email real time. It also supports sending attachments. The SMTP information (host, user name and password) may be provided or may be defined on a message sender, that may be provided as input. In addition, the business service supports using a default message sender defined as a [message option](#). Review the business service schema for information about the input elements.

NOTE: Validating attachments. If a Validate Email Attachment algorithm is plugged into the [installation record](#), it is called to validate the attachments supplied, if applicable.

NOTE: Retry setting. An option in the system properties file allows your implementation to configure the number of times to retry (if any) if the SMTP server is unavailable. Refer to the server administration guide for more information.

- Using an outbound message to send an email. This option allows for different variations as described in [Outbound Messages](#).
 - Some emails may be created en masse (for example a large group of emails routed to users for a given set of To Do entries). In this case, the records can be created in the staging table for processing using OSB.
 - Messages may still be sent real time using one of two business services described in [Real Time Messages](#). This option is an alternative to the dedicated email service described above when aspects of the outbound message functionality

are needed, such as the ability to instantiate a record as an audit or to include additional logic via BO plug-ins as part of sending the email.

JMS Message Browser

The JMS Message Browser portal allows you to select a JMS queue and view messages currently in the queue.

In order for a JMS queue to be available on the portal, a [message sender](#) must be defined that is configured for the appropriate JMS queue with the credentials to connect to the queue.

- If your organization sends real-time outgoing messages to a JMS queue, this configuration would exist as per the details in [Real-Time Message Configuration](#).
- If inbound web service messages are routed to the system via a JMS queue, no configuration is needed in the system. However, if you would like to view the messages in the queue in the JMS message browser portal, configuration for the JMS queues as described for outgoing messages is required.

Navigate to the portal using **Main > Integration > JMS Message Browser**.

The JMS Senders zone provides a list of configured message senders eligible for selection.

The JMS Message List zone is visible for the JMS Sender broadcast from the first zone. This zone supports selecting one or more records to delete from the queue. Use the message selector to limit the results to messages that satisfy the message selector. This uses standard JMS API message selector functionality. Refer to the zone inline help for information about the supported syntax.

The JMS Message Details zone displays a message broadcast from the list zone.

XML Application Integration

This section describes the XML Application Integration (XAI) utility, which enables you to configure your system to receive information from and to send information to external applications using Extensible Markup Language (XML) documents.

NOTE: The XAI functionality is legacy functionality and not recommended for new implementations. The topics for the functionality outlined in the previous sections describe the recommended features for supporting sending and receiving external messages. The XAI information remains in the documentation for upgrade purposes.

The Big Picture of XAI

The XML Application Integration (XAI) module provides the tools and infrastructure that businesses require for integrating their applications with your product. The integration your product with other systems across organizational boundaries or business is made possible, regardless of the platforms or operating system used. XAI provides an integration platform that enables the following:

- Integrate with Customer Relationship Management (CRM) systems
- Provide information feeds for web based customer portals
- Fit seamlessly with web based applications
- Facilitate fast implementation of batch interfaces
- Integrate with other XML compliant enterprise applications

XAI exposes system business objects as a set of XML based web services. The service can be invoked via different methods, e.g., Hypertext Transfer Protocol (HTTP) or Java Message Service (JMS). Consequently, any application or tool that can send and receive XML documents can now access the rich set of system business objects. Business-to-Business

(B2B) or Business-to-Consumer (B2C) integration with other enterprise applications as well as the setup of web portals is made very simple and straightforward.

XAI Architecture

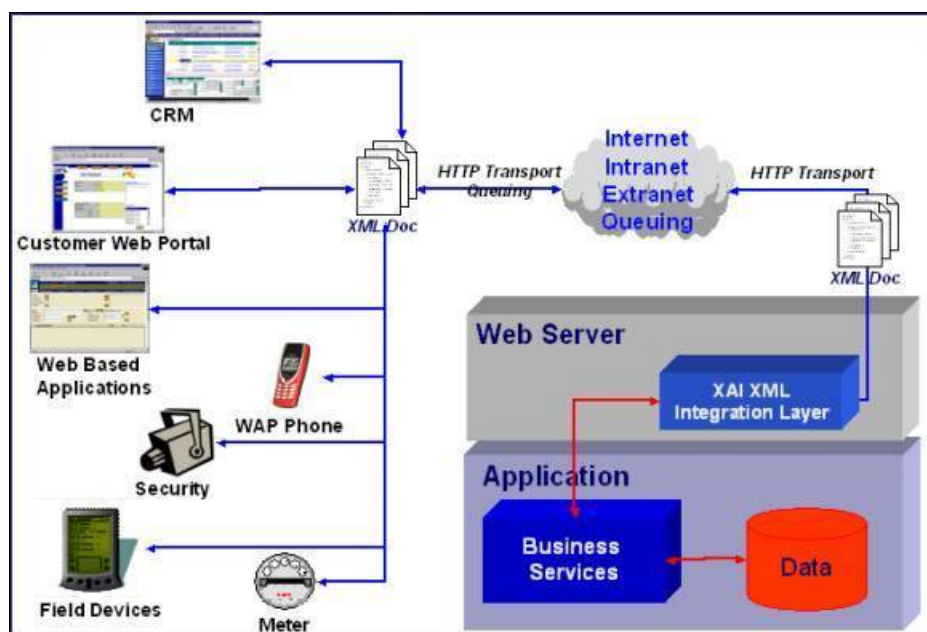
The XAI architecture contains 3 major components:

- The Core Server Component
- The Multi Purpose Listener (MPL)
- The Client Component

The Core Server Component

The core server component is responsible for receiving XML requests, executing the service and returning the response to the requester.

The following diagram shows the XAI tool operating on a web server and providing integration between the system business objects and various external applications.



The core is built in Java, using a layered, scalable architecture. The basic transport protocol supported by the core is SOAP/HTTP.

FASTPATH: Refer to [SOAP](#) for more information.

The XAI core server provides a Java servlet to process requests through HTTP. You may also use other messaging mechanisms such as message queuing, publish/subscribe or a staging table. The multi-purpose listener processes the messages.

The Multi Purpose Listener (MPL)

NOTE: Multi Purpose Listener functionality is legacy functionality that is not recommended going forward. The Oracle Service Bus (OSB) is the recommended tool. This section remains in place for upgrade purposes.

The Multi Purpose Listener (MPL) is a multi-threaded Java server that constantly reads XML requests from various external and internal data sources, such as a Java Message Service (JMS) message queue, a JMS topic or system staging tables.

The MPL can be used to process inbound messages (those sent by an external application to invoke a system service), or outgoing messages (those sent by your product to external applications). The MPL uses different receivers to process messages from different data sources.

A receiver is implemented using 3 distinct layers:

- The Receiving Layer
- The Execution Layer
- The Response Layer

The Receiving Layer

This layer deals with polling various locations to determine if new records, files or incoming requests exist. The various locations include:

- Staging tables, including [XAI staging control](#), [XAI upload staging](#), notification download staging (certain products only), and [outbound message](#).
- An external directory that contains a file, for example a comma delimited file or an XML file.
- A JMS queue/topic.

A separate receiver is defined to read requests from each of these locations. When the MPL server starts, it looks for all defined active receivers in the [XAI Receiver](#) table, and for each receiver it starts a thread that constantly fetches messages from the message source.

Once a request message is read, it is passed to an execution thread that implements the execution layer. Each receiver references an [Executer](#) that is responsible for executing the request.

Configuring Multiple MPL Servers

A single MPL server may only run one of each of the above staging table receivers for a given JDBC connection. To enhance the performance of the processing of the staging tables, you may define multiple MPL servers where each one runs the active receivers defined in the receiver table.

To ensure that each staging table receiver processes its own set of records in the staging table, the receiver selects a set number of records (specified as [Message option Number of Records an MPL Receiver Will Process At a Time](#)) and marks those records with the IP address and port number of the MPL.

The Execution Layer

The execution layer sends the XML request to the [XAI core server](#) and waits for a response.

NOTE: Currently the only type of [executer](#) supported is the XAI servlet running either on an XAI server or locally under the MPL. However the architecture allows for executing a request on other execution environments.

The executer is invoked and is passed in an [XAI Inbound Service](#) that specifies an XML request schema and an [adapter](#). Adapters tell XAI what to do with a request. The adapters point to a specific Java class that renders a service.

For example you can configure an Adapter to invoke any published application object (by pointing it to the appropriate java class). This adapter accesses system objects through the page service. You can think of an adapter as a plug-in component.

Once the executer processes the request and a response is received, it is transferred to the next layer, the [response](#) layer.

The Response Layer

The response layer is responsible for "responding" to the execution. The responses are handled by invoking an appropriate sender defined on the receiver's response information. Each sender defined in the system knows how to process its response. For example:

- The JMS queue sender and the JMS topic sender post responses to the appropriate queue / topic.
- The [staging control sender](#) handles errors received during the execution of the staging control request.

- The [upload staging sender](#) updates the status of the upload staging record based on the success or failure of the staging upload request.
- The download staging sender is a bit unusual because it is helping to build the message being sent (Oracle Customer Care and Billing only).

NOTE: There are some cases where a response is not applicable. For example, the file scan receiver creates a staging control record to process a file that exists in a directory. There is no "response" applicable for executing this request.

The XAI Client Component

The XAI Client component is the set of online control tables and tools used to manage your XAI environment.

The [Schema Editor](#) is a tool used to create and maintain XAI schemas.

FASTPATH: Refer to [XAI Schema](#) for more information.

The **Registry** is a term used to refer to all the tables required to "register" a service in the system. It includes the [XAI Inbound Service](#) and a set of control tables defining various options.

The [Trace Viewer](#) is installed with your XAI client tools and is used to view traces created on the XAI server.

XML Background Topics

The following section introduces some background information related to XML.

XAI Schemas

NOTE: Business Adapter. The **BusinessAdapter** adapter supports communication to configuration tool objects: [business objects](#), [business services](#) and [service scripts](#) through their own schema API. When communicating to these objects, it is not necessary to create XAI schemas for the schemas associated with the objects. As a result, there is no need to use the XAI schema editor when defining [XAI Inbound Services](#) for this adapter.

At the core of XAI are XML web services based on XML schemas. XML schemas define the structure of XML documents and are the infrastructure of e-business solutions. They are used to bridge business applications and enable transaction automation for e-commerce applications. Industry standard schemas document common vocabularies and grammars, enhancing collaboration and standardization. Validating XML processors utilize XML schemas to ensure that the right information is made available to the right user or application.

The system exposes its application objects as XML schemas that define the interface to system services. Every service (e.g., CreatePerson or AccountFinancialHistory) is defined using a pair of schemas: the Request Schema and the Response Schema. The request and response schema can be identical.

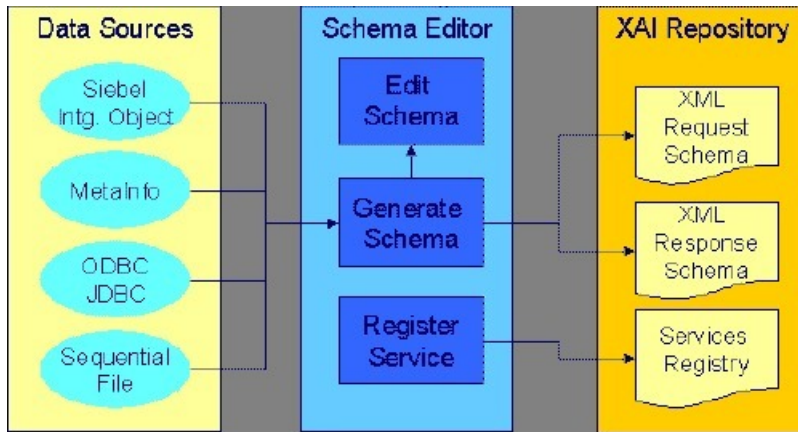
The Request Schema defines the XML document structure describing the "inputs" for a service.

The Response Schema defines the XML document structure describing the "outputs" of a service.

The Schema Editor

CAUTION: This schema editor is a separate legacy tool that is no longer recommended. The documentation remains for upgrade purposes

To facilitate the process of exposing business objects as XML schemas, we use the Schema Editor, a graphical tool to create, import and maintain schemas. The Schema Editor provides automated wizards to import schemas residing in existing data structures and documents. The Schema Editor can import schemas from the following sources: system business objects, ODBC data sources, sequential files.



Before the XAI tool can use a service, it must be registered or published.

FASTPATH: Refer to [Schema Editor](#) for more information.

XSL Transformations

XSL Transformations (XSLT) is a language used to transform an XML document into another XML document or another document format such as HTML. It is part of the Extensible Stylesheet Language (XSL) family of recommendations for defining XML document transformation and presentation. It is defined by the World Wide Web Consortium (W3C) and is widely accepted as the standard for XML transformations. Several tools are available on the market to generate XSLT scripts to transform an XML document defined by a source schema to an XML document defined by a target schema.

In XAI you can use XSL to:

- Transform an inbound message into the structure required by the XAI request schema for that service.
- Transform the response to an inbound message into a format defined by a schema provided by the requesting application.

FASTPATH: Refer to [XAI Inbound Service](#) for more information.

- Transform an outgoing message before it is sent out.

FASTPATH: Refer to [XAI Route Type](#) for more information.

- Transform data from an external source before it is loaded into the staging upload table.

FASTPATH: Refer to [XAI Inbound Service Staging](#) for more information.

SOAP

SOAP stands for Simple Object Access Protocol. The SOAP "Envelope" is the wrapping element of the whole request document that may be used by messages going through the XAI tool.

The following diagram shows a simple XML request using the SOAP standard.

```

<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <CorrelationID>1234</CorrelationID>
    <SOAPActionVersion>1.2</SOAPActionVersion>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <CISAccount transactionType='Read'>
      <CISAccountService>
        <CISAccountHeader
          AccountID='1234'
        />
      </CISAccountService>
    </CISAccount>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

XPATH

The XML Path Language (XPath) is an expression language used by XSLT to access or refer to parts of an XML document. It is part of the XSL recommendations maintained by the W3C. XPath provides the syntax for performing basic queries upon your XML document data. It works by utilizing the ability to work with XML documents as hierarchically structured data sets.

In the following XML document, some examples of XPath references are:

- authors/author/name
- authors/*/name
- authors/author[nationality]/name
- authors/author[nationality='Russian']/name
- authors/author[@period="classical"]

```

<?xml version='1.0'?>
<authors>
  <author period="classical">
    <name>Sophocles</name>
    <nationality>Greek</nationality>
  </author>
  <author>
    <name>Leo Tolstoy</name>
    <nationality>Russian</nationality>
  </author>
  <author period="classical">
    <name>Plato</name>
    <nationality>Greek</nationality>
  </author>
</authors>

```

In the XAI tool, XPath is used to construct outgoing messages.

Server Security

XAI server security supports the basic HTTP authentication mechanism as well as web service security (WS-Security) to authenticate the user requesting service. When authenticating using WS-Security, the SOAP header contains the authenticating information.

The base package provides two XAI server URLs, one that uses basic HTTP authentication ('/classicxai') and another that supports both methods ('/xaiserver'). Regardless of which authentication method is practiced, it is the latter you should

expose as your main XAI server. The main XAI servlet gathers authentication information from the incoming request (HTTP or SOAP header) and further calls the internal ("classic") servlet for processing.

The "classic" XAI server security uses the basic HTTP authentication mechanism to authenticate the user requesting service. It assumes the requester has been authenticated on the Web server running the XAI servlet using the standard HTTP (or HTTPS) basic authentication mechanism. The authenticated user-id is passed to the application server, which is responsible for enforcing application security. This requires the system administrator to enable basic authentication for the Web server running the XAI servlet. To enable HTTP basic authentication, the XAI server '/classicxai' should be defined as a url-pattern in the web resource collection in the web.xml file. When the XAI server is not enabled for basic authentication, it transfers the user-id specified on the **Default User** [Message Option](#) to the application server.

By default, the system would always attempt to further authenticate using SOAP header information. This is true even if the request has already been authenticated via the Web server. Use the **Enforce SOAP Authentication** [Message Option](#) to override this behavior so that a request that has been authenticated already by the Web server does not require further authentication by the system.

If SOAP authentication information is not provided, the system attempts to authenticate this time using information on the HTTP header. You can force the system to solely use SOAP authentication using the **Attempt Classic Authentication** [Message Option](#).

Currently the system only supports the standard *Username Token Profile* SOAP authentication method where "Username", "Password" and "Encoding" information is used to authenticate the sender's credentials. The following is an example of a *Username Token Profile* in a SOAP header:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "urn:schemas-xmlsoap-org:envelope">
<SOAP-ENV:Header xmlns:wsse="http://www.w3.org/2001/XMLSchema-instance">
<wsse:Security>
<wsse:UsernameToken>
<wsse:Username>MYUSERID</wsse:Username>
<wsse:Password Type="PasswordText">MYPASSWORD</wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
<SOAPActionVersion>2.0.0</SOAPActionVersion>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
...
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

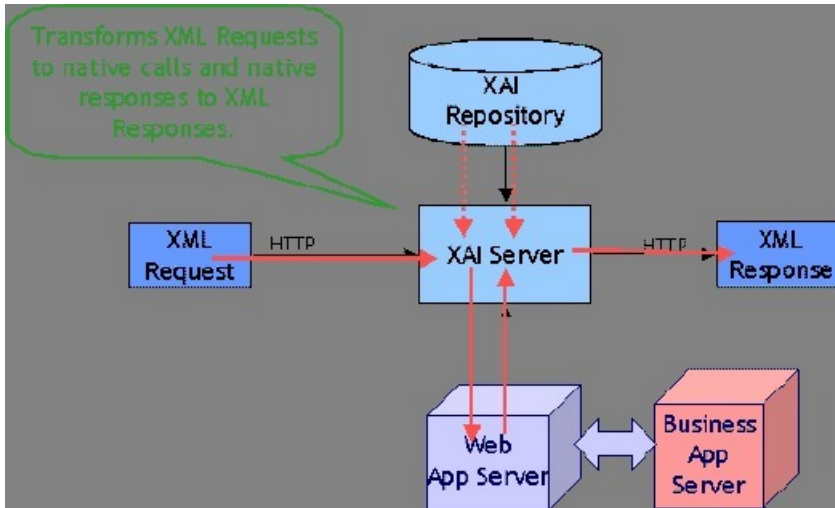
By default both user and password are authenticated. You can use the **System Authentication Profile** [Message Option](#) to change this.

NOTE: Custom authentication. You can override the base package user credentials authentication logic using the **System Authentication Class** [Message Option](#).

Inbound Messages

Inbound messages are XML messages sent by an external application to invoke a system service. Inbound messages can be sent one at a time or in batches of several messages in a file. Third party integration points can import web services for inbound service calls. The standard method of doing this is by publishing a WSDL (Web Service Definition Language) definition for each exposed service.

Synchronous Messages



Synchronous service requests are sent using the HTTP protocol to the XAI servlet, which runs under a web application server such as WebLogic.

- The service request XML document must conform to the request schema that defines the service.
- The XAI servlet on the web server receives the service request XML document and based on the service name in the document identifies the appropriate XAI Inbound Service. Once the service is identified, the XAI servlet accesses the XAI Inbound Service record to find out the properties of the service.
- Based on the service properties, the XAI module loads the request and response schemas from the schemas repository (and caches them in memory).
- Based on the Adapter referenced by the service, it calls the appropriate adapter. The adapter performs most of the work to service the request.
 - The adapter connects to the application server, based on the connection information in the registry control tables.
 - It then parses the request document using the request schema.
 - Once the request document has been validated, the adapter converts the XML request document into a call to the application server.
- The response returned by the application server is then converted into a service response XML document, based on the response schema.
- The XML response document is shipped back to the caller over HTTP.

Using HTTP for Synchronous Service Execution

Invoking a service is not much different from sending a regular HTTP request. Here the HTTP request contains the XML as a parameter. The XAI server handling requests via the HTTP protocol is implemented using a Java servlet running on the web server.

Microsoft Visual Basic/C Example

Microsoft provides an easy way to send XML requests over HTTP. To send and receive XML data over HTTP, use the Microsoft XMLHTTP object

```
set xmlhttp = createObject("Microsoft.XMLHTTP")
xmlhttp.Open "POST", http://localhost:6000/xaiserver, false
xmlhttp.Send XMLRequest
```

```
XMLResponse = xmlhttp.ResponseText
```

Here **http://localhost:6000/xaiserver** is the URL of the XAI server. **XMLRequest** contains the XML request to be processed by XAI and **XMLResponse** contains the XML response document created by the XAI runtime.

Java Example

Java provides a very simple way to send a request over HTTP. The following example shows how a request can be sent to XAI from an application running under the same WebLogic server as the one XAI runs. In this example, we use the "dom4j" interface to parse the XML document.

```
import com.splwg.xai.external.*;
import org.dom4j.*;

String xml;
xml = "<XML request>";
XAIHTTPCallForWebLogic httpCall = new XAIHTTPCallForWebLogic();
String httpResponse = httpCall.callXAIServer(xml);
Document document = DocumentHelper.parseText(httpResponse);
```

Asynchronous Messages

Various types of [receivers](#) running under the MPL server (rather than the XAI server) handle asynchronous inbound messages from several sources.

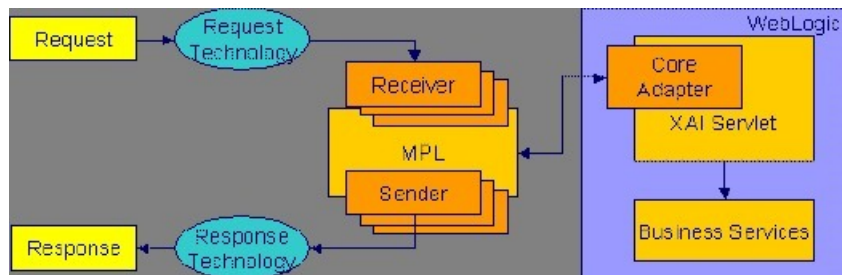
Requests may be received via the following:

- MQSeries, TIBCO or any JMS compatible messaging system
- The XAI Upload Staging table

The response is returned to the JMS queue/topic or to the staging table.

FASTPATH: Refer to [Designing XAI Receivers](#) for more information about the different receivers provided by the product for the different data sources.

The following diagram shows the flow for asynchronous inbound messages.

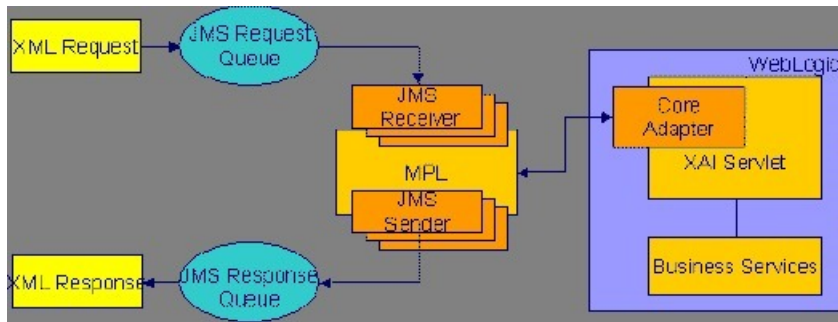


Using JMS

Java Message Services (JMS) is a standard Java 2 Platform, Enterprise Edition (J2EE) protocol to send/receive asynchronous messages using a queue or topic message protocol.

XML messages may be received and sent via JMS using either a JMS Queue or JMS Topic. In order to access a JMS provider such as MQSeries, TIBCO or WebLogic, the MPL must first connect to the appropriate server using a JMS Connection.

The following diagram depicts a message sent and received through a JMS Queue.



Using JMS Queues

JMS Queues are used to receive and send messages using the message queue protocol. Products that support this protocol include MQSeries.

The following describes events that take place when a JMS queue is used:

- The requester places the XML request message on a JMS queue. The request contains both the XML message and the name for the *reply queue*.
- The **JMS Queue Receiver** waits for messages on that queue. When the message arrives it is selected from the queue and executed using the adapter for the requested service.
- The XML response message is placed on the reply queue specified in the request. It is the requester's responsibility to fetch the response message from the queue.

The MPL uses a **JNDI server** to locate the queue resource.

Using JMS Topics

JMS Topics are used to send and receive messages using the publish/subscribe messaging model. Products that support this protocol include TIBCO, MQSeries and WebLogic.

The MPL uses a JNDI server to locate the topic resource.

- Define a **JMS Topic receiver**, listening to a predefined JMS Topic.
- The other application builds an XML message based on the schema defined for that service.
- It sends the XML request to the predefined topic and specifies the reply topic name.
- The MPL reads the message from the Topic, executes the service and returns the response to the reply topic specified in the inbound message.

Staging Upload

The system provides a staging table, where an interface can store XML requests to perform a service in the system.

Some external systems interfacing with the system are not able to produce XML request messages. Or you may have external systems that produce XML messages but the messages are sent in a batch rather than real time. The system provides the capability to read an external data source containing multiple records, map the data to an XML request and store the request on the **XAI upload staging** table. These records may be in XML requests, sequential input files or database tables.

The XAI upload staging table may be populated in one of the following ways:

- When a collection of messages in a file or database table must be uploaded, a **staging control** record should be created for each file/database table. The **Staging Control Receiver** processes each file or database table and creates records in the XAI upload staging table for each message.
- The **XML File Receiver** creates records directly in the XAI upload staging table. Note, in addition, the XML File Receiver creates a staging control record to group together these records.
- XAI creates records in the XAI upload staging table for **inbound messages in error** that are configured to post the error.

- It is possible that when a response to a notification download staging message is received (Oracle Customer Care and Billing only), the response requires some sort of action. If this is the case, the system creates an XAI upload staging record (and an XAI staging control record) for the response.

Staging Upload Receiver

Once the XML requests are in the staging table, the [Staging Upload Receiver](#) reads the requests from the XAI upload staging table and invokes the XAI server (via the executor) with the appropriate XAI inbound service. Inbound service records typically point to the core [adapter](#) used to invoke system services.

Staging Upload Sender

The staging upload sender handles "responses" to the execution of the message in XAI upload staging. If the execution is successful, the sender updates the status of the upload staging record to **complete**. If the execution is unsuccessful, the sender updates the status to **error** and creates a record in the [XAI upload exception](#) table.

NOTE: Configuration required. The above explanation assumes that you have correctly configured your upload staging receiver to reference the upload staging sender. Refer to [Designing Responses for a Receiver](#) for more information.

Staging Control

The [staging control](#) table is used to indicate to XAI that there is a file or table with a collection of records to be uploaded. The special [Staging Control Receiver](#) periodically reads the staging control table to process new records.

The XAI staging control table may be populated in one of the following ways:

- To process a specific sequential input file, manually create a staging control record and indicate the location and name of the file and the XAI inbound service to use for processing. Use this mechanism to process ad-hoc uploads of files.
- If a file is received periodically, you may define a [File Scan Receiver](#), which periodically checks a given file directory for new files. The file scan receiver creates a new staging control record to process this file.
- The [XML File Receiver](#) processes a file containing a collection of XML messages to be uploaded. The XML file receiver creates a staging control record and creates records directly into the XML upload table. The staging control record is automatically set to a status of **Complete** and is used to group together the XML upload records. Refer to [Processing Staging Upload Records for a Staging Control](#) for more information.
- To upload records from a database, manually create a staging control record and indicate an appropriate XAI inbound service, which contains the information needed by the system to access the appropriate table. Use this mechanism to process ad-hoc uploads of files.

NOTE: To upload records from a database table, you must create a staging control record. There is no receiver that periodically looks for records in a database table.

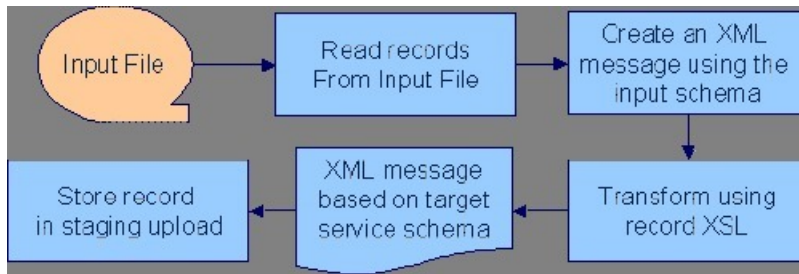
- Whenever an [XAI upload staging](#) record is created, an XAI staging control record is created as well.

FASTPATH: Refer to [Batch scenarios](#) for more information about configuring the system to populate the staging control with requests from external files.

Staging Control Receiver

The Staging Control [Receiver](#) processes staging control records and invokes XAI (via the executor) to execute the request. The executor uses the appropriate adapter to generate records in the [XAI Upload Staging](#) table - one for each record in the file or table.

The diagram below illustrates the information used by the staging control receiver to load data onto the staging table from a sequential input file.



The staging control [adapter](#) does the actual work. It reads the individual records in the input file and applies the XSL transformation script indicated on the XAI inbound service record to the input data to produce an XML request in the XAI upload staging table.

Processing Staging Upload Records for a Staging Control

In some cases, a process may populate records directly into the XML staging upload table. An example of such a process is the [XML File Receiver](#). In this case, a staging control record is also created and used to group together the staging upload records.

The staging control contains information needed to process a group of staging upload records:

- The user related to these records. This user is for application security purposes. The user indicated here must have the proper rights for the application service and transaction type to be executed by XML upload records processed for this staging control.
- An indication of whether or not the records should be processed [sequentially](#).

Processing Staging Records in Sequential Order

In some cases, a collection of messages uploaded together in a file must be processed in the order the messages are received. For example, if messages to add a person and add an account for this person are received together, the message to add the person must be processed before the message to add the account.

If messages received in a file must be processed sequentially, turn on the Sequential Execution switch on the [staging control](#) record. When the staging control receiver creates records in the XML upload table, the identifier of each record is built as a concatenation of the staging control record and a sequential number. If your staging control record indicates that the XML upload records should be processed in sequential order, the records are processed in primary ID order.

NOTE: Non-sequential processing. If your staging control does not indicate that the related XML records should be processed in sequential order, the records are processed by the staging control receiver using a random, multi-threaded mechanism.

If you have defined a [receiver](#) to periodically search for files and populate records in the staging control table, you may turn on the Sequential Execution switch on the receiver. This ensures that records processed as a result of this receiver are executed in sequential order.

Staging Control Parameters

If your staging control accesses the data from a database table, you have the capability of defining the selection criteria. [XAI inbound services](#) that reference the **CISStagingUpload** adapter may contain a collection of fields that are used in an SQL WHERE clause. When adding a new [XAI Staging Control](#) record, you can define the values for the WHERE clause.

For example, imagine that you have a work management system where new premises are defined. Rather than waiting for this system to "push" new data to you, you design the interface to have the system "pull" the new data by looking directly at the "new property" database table.

The Request XML for your XAI service contains SQL statements used to access the data. You could define a Staging Control Parameter of "Add Date". When creating your staging control, you may enter a parameter value of today's date. When this record is processed, it only retrieves new properties from this work management table whose Add Date is today.

The following example shows part of the Request XML schema for an XAI Service that SELECTs premises based on postal code.

```

- <Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes" xmlns:d="urn:schemas-microsoft-com:datatypes" xmlns:b="urn:schemas-microsoft-com:BizTalkServer" b:root_reference="TestDuplicatePremise">
- <ElementType name="TestDuplicatePremise" content="eltOnly">
  <b:property name="adapter">CISStagingUpload</b:property>
  <b:property name="stagingUploadType">DB</b:property>
  <element type="Request" />
  <element type="Response" />
</ElementType>
- <ElementType name="Request" content="eltOnly">
  <AttributeType name="Postal" d:type="string" d:maxLength="12" />
  <b:RecordInfo />
  <attribute type="Postal" />
</ElementType>
- <ElementType name="Response" content="eltOnly">
  <b:RecordInfo />
  <element type="Premises" />
</ElementType>
- <ElementType name="Premises">
  <b:RecordInfo />
  <b:property name="SQL">SELECT PREM_TYPE_CD, KEY_SW, OK_TO_ENTER_SW,
TREND_AREA_CD, ADDRESS1, MAIL_ADDR_SW, POSTAL FROM CISADM.CI_PREM WHERE
POSTAL = @TestDuplicatePremise/Request/Postal</b:property>
  <b:property name="tableName">CISADM.CI_PREM</b:property>
- <AttributeType name="PREM_TYPE_CD" d:type="string" d:maxLength="000">
  <b:property name="dataType">string</b:property>
  <b:property name="uniqueId">PREM_TYPE_CD</b:property>
  <b:FieldInfo />
</AttributeType>
- <AttributeType name="KEY_SW" d:type="string" d:maxLength="001">
  <b:property name="dataType">string</b:property>
  <b:property name="uniqueId">KEY_SW</b:property>

```

The postal code value to substitute into the WHERE clause is defined on the individual XML Staging Control records.

FASTPATH: Refer to [XAI Staging Control](#) for more information.

Staging Control Sender

Before the staging control receiver invokes the executer, it changes its status to **complete**, assuming that there will be no problems. If the executer detects an error condition, the staging control sender updates the status of the staging control record to **error** and removes any XAI upload staging records that may have been created.

NOTE: Configuration required. The above explanation assumes that you have correctly configured your staging control receiver to reference the staging control sender. Refer to [Designing Responses for a Receiver](#) for more information.

Inbound Message Error Handling

For messages that are processed using the [staging upload](#) table, application errors that prevent XAI from successfully processing the message cause the staging record to be marked in error and highlighted via a To Do entry.

For messages that are not processed via staging upload, your implementation should consider what should happen to application errors. If the origin of the message is able to handle an immediate error returned by XAI, then no special configuration is needed. An example of this is an HTTP call to our system where the originator of the message is waiting for a real-time response.

Otherwise, for messages where errors should not be returned to the originator, but should be highlighted in this system for resolution, be sure to mark the **Post Error** switch on the [XAI inbound service](#). When this switch is turned on and an application error is received by XAI when processing the message an [XAI upload staging](#) record is created (along with a staging control record) and marked in **error**.

For example, if a message is received via a JMS queue, application errors that prevent XAI from processing the message should not be returned to the queue because there is no logic to route the error to the sending system.

Integration Scenarios

Integration Using an EAI (or Hub)

It is possible for your various systems to be integrated with each other using a hub. The hub is implemented using an Enterprise Application Integration (EAI) tool provided by a third party vendor. Most hubs support HTTP and/or JMS and can work with XML schemas or document type definitions (DTDs).

- XAI services are presented to the EAI tool as schemas or as DTDs, immediately making the system callable from the hub.
- Integration scripts or workflow processes are defined in the EAI tools.
- At run time the hub uses HTTP or JMS to access the system using inbound messages.
- Outgoing messages are used to notify the hub about events occurring in the system. The messages are sent using HTTP or JMS.

Batch Scenarios

Messages may be sent in batch files, or may be retrieved from a database. In all cases, the system needs to be able to read the file and identify each individual message in order to create an XML request that can be processed by the XAI server. Once each individual message is identified, a request is stored on the XAI Staging Upload table for later execution.

XML Message File

It is possible for you to receive a file containing a collection of XML messages. The system identifies each separate message within the file and creates an entry for each message on the XAI upload staging table. It also creates a staging control record to group together each newly created XAI upload record. This staging control is created in **Complete** status and is not processed by the staging control receiver.

Since external applications may send messages in a format unknown to XAI, the system needs a mechanism for identifying the messages and mapping them to an XAI service.

XAI Groups

First the system associates the entire XML file with an XAI Group. You can think of the XAI Group as a categorization of the collection of messages. For example, you may have a separate XAI Group for each third party who sends you a collection of XML messages.

The system uses an [XPath](#) and XPath value to identify the correct XAI Group for the XML file.

Attachments and Rules

After identifying the appropriate group to which an XML file belongs, the system takes each message in the file and applies the appropriate XSL transformation to the message to produce a record on the upload staging table.

To process the messages in a file, the system needs to know how to identify each message in a file containing multiple messages. A file may use the same root element for each message or different root elements for different types of messages. For each XAI group, you must indicate the root element(s) that identifies a message by defining one or more attachments. Each attachment defines a root element, which tells the system when a new message begins.

Once the system has identified each separate message in the file, it must determine the correct XSL transformation script to apply. Once again the system uses an [XPath](#) and XPath value to identify the correct XSL to apply. For each XAI group, you define one XAI rule for every possible type of message you may receive in the file. Each XAI rule defines an XPath, XPath value and XSL transformation script.

Note you may assign a priority to each of your rules. The rules for more common messages may be assigned a higher priority. This enhances performance by ensuring that rules for more common messages are processed before rules for less common messages.

NOTE: Include parent elements. If the XML message includes parent elements (such as a transmission id or a date/time) that are needed for any of the separate child messages that are posted to the upload staging table, you can configure the appropriate attachment to include **parent** elements.

FASTPATH: Refer to [Designing an XML File Receiver](#) for more information about defining receivers that process XML files.

Sequential Input File

You may receive messages in a sequential input file, such as a comma-delimited file.

The following steps should be performed when configuring the system to enable data to be uploaded from an input file into the staging upload table:

1. Create an XML Schema that describes the records in the sequential input file.
2. Create the XSLT transformation that maps a record in the input file to an XML service request in your product.
3. Create an [XAI service](#) representing the batch process that loads the input file into the staging table.
4. If desired, create a new file scan receiver, which waits for an input file to appear in a particular directory. (If you do not take this step, then you need to create a [staging control](#) when you want a file to be processed.)

NOTE: Character Encoding. If the file is encoded with a specific character encoding, you may indicate the encoding as part of the file name to be uploaded. If the file name ends with **?enc=?x**, where x is the file character encoding, the adapter processes the file accordingly. For example, the file name may be specified as **premiseUpload.csv?enc=?UTF-8**. If the encoding is not specified as part of the file name and the file is in UTF-16 or UTF-8 with byte order mark, then the adapter can recognize the encoding.

Database File

It is possible for you to define an interface where inbound messages are retrieved by reading records in a database table.

The following steps should be performed when configuring the system to enable data to be uploaded from a database table into the staging upload table:

- Create an XML Schema that describes the records in the database table.
- Create the XSLT transformation that maps a record in the database table to an XML service request in your product.
- Create an [XAI service](#) representing the process that loads the records from the database table into the staging table.

WSDL Catalog

Web Service Definition Language (WSDL) is a language for describing how to interface with XML-based services. It acts as a "user's manual" for Web services, defining how service providers and requesters communicate with each other about Web services.

The base package provides the ability to publish a WSDL definition for each service exposed as an [XAI Inbound Service](#). In addition, it is possible to request a catalog of all the XAI Inbound Services and a link to each WSDL. To view the catalog, launch a new browser session and enter the URL **http://\$host:\$port/XAIApp/xaiserver?WSDL** where \$host and \$port are replaced by the appropriate values for the current environment.

Outgoing Messages

This section describes outgoing message functionality related to MPL, which is no longer recommended.

- Outbound messages. As described in [Outbound Messages](#), outbound messages are supported for sending outgoing messages. This functionality includes support that is only related to MPL.
- Notification download staging (NDS) messages. This method is only supported by *Oracle Utilities Customer Care and Billing*. Using this method near real-time, a record is written to the NDS staging table referencing only key fields. MPL then polls the records, invokes a service to build the message, applies the XSL and routes the message. If sent real-time, no record is posted to the staging table but rather the message dispatcher routes the message immediately. Refer to Oracle Utilities Customer Care and Billing help, Workflow and Notifications, Notification and XAI for more information.

The following sections describe the outbound messages topics specific to MPL in more detail.

Outbound Message Receiver

The outbound message [receiver](#) processes records in the outbound message table that have a processing method flag equal to **XAI** and a status of **pending** and changes the status to **in progress**. The receiver then retrieves the message XSL and the message sender defined for the external system / outbound message type.

NOTE: Template external system. If the outbound message's external system references a template external system, the outbound message type configuration for the template external system is used.

It applies the message XSL (if supplied). If the option to [validate outbound message schemas](#) is turned on, the schema validation is performed.

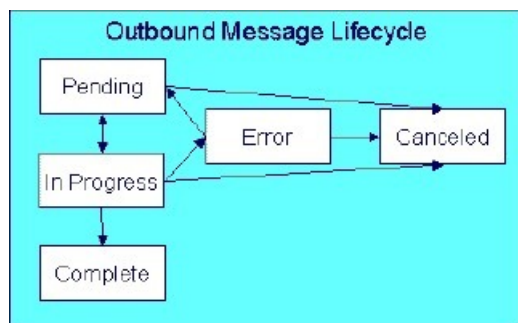
Refer to [Outbound Message Error Handling](#) for information about error handling.

If no errors are received, control is turned over to the [outbound message sender](#) for routing.

NOTE: Initialization of receiver. During the initialization of this receiver (for example if there is a problem with MPL and it is restarted) any records that are found to be **in progress** are changed to **pending** so that those messages are sent properly.

Lifecycle of Outbound Message

The outbound message receiver processes outbound message records based on their status. The following diagram describes the lifecycle of an **XAI** type outbound message.



- Records are created in **pending** status.
- The outbound message receiver processes pending records and changes the status to **in progress**.
- If the message is sent successfully the system changes the status to **complete**.
- If there was a problem sending the message the system changes the status to **error**.
- When the user resolves the error they can change the status back to **pending**.
- A user can change the status of a **pending** or **error** record to **anceled**.
- For the rare cases where there is a problem with MPL and a message is left in the status **in progress**, users may manually change the status to **anceled**. In addition the outbound message receiver includes a step at startup to find **in progress** messages and change them to **pending**.

Outbound Message Sender

The outbound message sender is responsible for routing the message to the [Message sender](#) determined by the receiver. If the routing is successful the outbound message status is marked **complete**. If the routing is unsuccessful, the status is marked in **error**.

Refer to [Outbound Message Error Handling](#) for information about error handling.

NOTE: Automatic Resend. If you have configured the system for [automatic resend](#) and the system detects that the error is due to the sender being unavailable, the message remains in **pending** status.

NOTE: Configuration required. The above explanation assumes that you have correctly configured your outbound message receiver to reference the outbound message sender. Refer to [Designing Responses for a Receiver](#) for more information.

Outbound Message Error Handling

If the outbound message receiver or the outbound message sender detects an error while attempting to process the outbound message, it marks the message in **error**, captures the error message and its parameter values and creates a To Do entry using the To Do type specified in the Message option **To Do Type for Outbound Message Errors**.

A separate background process [F1-DTDOM](#) is responsible for completing To Do entries for outbound messages no longer in **Error**.

Automatic Resend

If a system error is received by the MPL when attempting to route the message to a sender, (using the outbound message method or the NDS message method), the system marks the appropriate table in **error**. This is true even if the reason for the error is that the connection to the sender is unavailable. When the connection is restored, a user must change the status of the appropriate record to **pending** (for outbound messages) or **retry** (for NDS messages) in order for the message to be resent.

Alternatively, you can configure your system to attempt to automatically resend the message. This section describes the logic available for auto resend. To enable automatic resend, you must set the flag **Automatically Attempt Resending to Unavailable Senders** on [Message option](#) appropriately.

If an error is received by the MPL when it attempts to invoke a sender and the auto resend option is on, the system does not mark the record in **error**. It continues to attempt sending messages to the sender until the number of errors has reached a predefined maximum error number (defined as an [Message option](#)). When the maximum is reached, the sender is marked as **unavailable** and an MPL log entry is created. The MPL ignores messages for **unavailable** senders.

The system tries to resend messages to this sender the moment the sender is reset to be **available**. The following points describe how a sender becomes **available**:

- MPL attempts to retry sending messages to unavailable senders every X minutes, where X is defined on [Message option](#).
- On MPL startup all senders are marked as **available**
- A user may navigate to the [XAI Command](#) page and issue a command **MPL Refresh Sender** to refresh the cached information for a particular sender

Designing Your XAI Environment

This section guides you through the steps required to design the tables that control your XAI processing.

Installation

The XAI server is installed with default configuration. This section describes how you may customize the XAI server configuration.

Startup parameters are defined in two parameters files

- The XAIParameterInfo.xml file is used by the XAI HTTP server. This file is found within the XAI directory in the path (...\\splapp\\xai).
- The MPLParameterInfo.xml file is used by the MPL server. This file is found within the XAI directory in the path (...\\splapp\\mpl).

Both files store the parameters as XML files with the following elements (sections):

- Source
- ParameterVariables
- AdHocParameters

The XAI Source Section

The XAI tool accesses XAI [registry](#) information through the standard system programs. The <Source> section in the XAIParameterInfo.xml file tells XAI the user ID for accessing the registry information. It contains the following attributes:

Attribute Name	Description
Source Type	This should be set to CorDaptix , which tells XAI to access the registry through the standard access to system programs.
CorDaptixUser	The user ID to use when accessing the registry data.

The MPL Source Section

The <Source> section in the MPLParameterInfo.xml file defines the database connection information used to connect to the database storing the XAI table information. It contains the following attributes:

Attribute Name	Description
Source Type	Defines the source of the data, for example ORACLE or DB2 .
jdbcURL	The URL used to connect to the product database. For example: jdbc:oracle:thin:@//server-name:1234/DBNAME
databaseUser	The Oracle User Id used to connect to the database.
databaseUserPassword	The Oracle password used to connect to the database.

The Parameter Variables Section

When defining values for fields in certain control tables in the registry, you may reference substitution variables that point to the <ParameterVariables> section of the installation files. Substitution variables provide for dynamic substitution of values based on parameters provided at server startup.

To specify a substitution parameter in a string value you enter the name of the substitution parameter enclosed with @.

For example if you have a field in the XAI control tables that should contain the URL for the XAI HTTP servlet, you could enter the value in the following way: **http://@HOST@:@PORT@/xaiserver**.

In the parameters section, define the appropriate values for these parameters, for example:

```
<ParameterVariables>
<ParameterVariable name="HOST" value="localhost" />
<ParameterVariable name="PORT" value="8001" />
</ParameterVariables>
```

At run time, the system builds the URL as **http://localhost:8001/xaiserver**.

Every substitution parameter is defined using an <ParameterVariable> element with the following attributes:

Attribute Name	Description
Name	The name of the substitution parameter
Value	The value to replace an occurrence of the substitution parameter in an XAI control table field

NOTE: Substitution parameters can only be used for string fields, and not for fields that are foreign keys to other objects.

The AdHoc Parameters Section

The <AdHocParameters> section is used to provide registry definitions that override the existing ones. Unlike the <ParameterVariables> section, a whole registry object definition can be specified in this section. When the XAI server starts, it first reads the registry definitions from the database and then it reads the <AdhocParameters> section. If it finds an object definition in this section, it uses it to replace the one read from the database.

Attribute Name	Description
Object Name	The object name may be one of the following objects: Option Receiver Sender
Object Attributes	The attributes of the object. Each object type has its own set of attributes:
'Option' object attributes	
name	The option flag. Must be defined in the OPTION_FLG table
value	The value for that option
'Receiver' object attributes	
name	The receiver ID
Class	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries
JMSProvider	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries
Executer	The XAI Executer ID for this receiver
'Sender' object attributes	
Class	The JMS provider. May be 'MQ'
JMSProvider	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries

Designing XAI Inbound Services

When designing your XAI environment, you should first identify the services that you would like to perform. Determining your services facilitates your design for the other registry options.

To design your inbound services,

- Determine each service that needs to be performed
- Determine the correct [adapter](#) that is needed by your service.
- Determine the required layout of the request and response messages and specify the request [schema](#) and response [schema](#).

- If a transformation of the data is required, you need to design the appropriate request XSL and response [XSL transformation](#) scripts.
- If the service references the staging upload adapter, determine the staging file type and design the record XSL transformation script. In addition, determine if you want to enter an input file name and interface name. Finally, determine whether or not you need to indicate a special [JDBC connection](#).
- As new releases of the system are installed, it may be necessary to modify your service for the new release. If this is the case, you need to design separate versions of the inbound service.

FASTPATH: Refer to [XAI Inbound Services](#) for more information.

Designing XML Schemas

You need XML schemas for the services you designed in [Designing XAI Inbound Services](#).

For each message, identify what service you need to invoke and what action you need to perform. If you have multiple actions that you may need to perform for the same service, you may choose to create a single generic XML schema or you may choose to create multiple schemas, which are more specific. For generic messages, the transaction type, indicating the action to perform would be passed in on the XML request document to indicate what must be done. For more specific messages, you may be able to indicate the transaction type directly on the schema and it would not need to be overwritten at run time.

You need to create a response schema for each request schema. It is possible for you to use the same schema for both functions.

FASTPATH: Refer to [Schema Editor](#) for more information.

Designing XSL Transformations

You need an XSL transformation script for each service you designed in [Designing XAI Inbound Services](#), where you determined a transformation is necessary. In addition, you need XSLT scripts for your outgoing messages. Each sender, which receives a message, probably requires a transformation of the message into a local format. Refer to [Outgoing Messages](#) for more information.

For each message requiring transformation, determine the format used by the external system. In most cases, it is not the same format recognized by the system. For each case, you must create an XSL transformation, which maps the message format from the external format to one expected by your product or from your product format to one expected by the external system.

When identifying the required XSL transformations, remember to take into consideration the data that is processed by the staging control table. This service reads data stored in a file or database table and uses the Record XSL to map the individual records to an individual service request.

FASTPATH: Refer to [XAI Inbound Service](#) for more information.

Designing Your Registry Options

The XAI registry is a set of control tables that is used to store service definitions as well as various system information required by the XAI and MPL servers. The following sections describe each table in the registry.

Designing XAI JDBC Connections

If you need to access a database table to process your messages, XAI needs to know the location of the database and how to access it. If the tables are located in the same database used for the system (defined in your [Installation](#)), then you do not

need to enter any extra JDBC Connections. If you need to access data that lives in another database, design the additional JDBC Connections and determine the type of connection and connection information.

FASTPATH: Refer to [XAI JDBC Connections](#) for more information about defining XAI JDBC Connections.

Designing XAI Formats

The Formats section of the registry is used to define data formats. Data formats can be used in schema definitions to specify data transformations. To determine what data formats you need to define for your XAI environment, you must review the expected format of data that you will be exchanging and determine whether or not data transformation is required.

The following sections describe the four different types of formats and some guidelines in their use.

Date Formats

Date formats may be specified using any valid Java format supported by the `java.text.SimpleDateFormat` class.

To specify the time format use a time pattern string. For patterns, all ASCII letters are reserved. The following usage is defined:

Symbol	Meaning	Presentation	Example
G	era designator	Text	AD
y	year	Number	1996
M	month in year	Text & Number	July & 07
d	day in month	Number	10
h	hour in am/pm (1-12)	Number	12
H	hour in day (0-23)	Number	0
m	minute in hour	Number	30
s	second in minute	Number	55
S	millisecond	Number	978
E	day in week	Text	Tuesday
D	day in year	Number	189
F	day of week in month	Number	2 (2 nd Wed in July)
w	week in year	Number	27
W	week in month	Number	2
a	am/pm marker	Text	PM
k	hour in day (1-24)	Number	24
K	hour in am/pm (0-11)	Number	0
z	time zone	Text	Pacific Standard Time
'	escape for text	Delimiter	
"	single quote	Literal	'

Currency Formats

Currency formats are used to specify formatting for elements representing currencies. They may include the following:

Symbol	Meaning
#	number place holder
,	thousands separator
.	decimal point
\$	currency sign

For example to define the currency format for US dollar, indicate: `$$,#.00`

Phone Formats

Phone formats can be used to specify formats for telephone numbers. The supported format specification is limited to the following format characters:

Symbol	Meaning
0	number place holder
\0	0

Any other character appearing in the formatting expression is a placeholder for that character. To specify the '0' character, use '\0'.

Phone Format Example: (000) 000-0000

Text Formats

Text formats are used to specify formats for character string attributes. The following expressions are supported:

Symbol	Meaning
\cUpperCase	Translate the string to upper case.
\cLowerCase	Translate the string to lower case.
\cProperCase	Translate the string to proper case. The first character of every word is translated to uppercase. The remaining characters are translated to lowercase.

FASTPATH: Refer to [XAI Format](#) to define your XAI Formats.

Designing XAI Adapters

The product provides a set of adapters to process your XML requests. The adapters point to a specific Java class that renders a service. If you find that you need to use a protocol, which is not supported by the adapters provided, you will need to add a new Message Class (which points to a Java class) and a new XAI Adapter. It is recommended that your implementers contact customer support. The following adapter classes are provided.

- **BUSINESSADA:** This is the adapter class that provides access to schema-based objects. This adapter accesses [business objects](#), [business services](#) and [service scripts](#) through their schema API. Services with this adapter need to indicate the schema of the object, which should be invoked. When communicating to these objects, it is not necessary to create XAI schemas for the schemas associated with the objects. XAI is able to directly communicate with these objects using their existing schema definitions. As a result, there is no need to use the XAI schema editor when defining XAI Inbound Services for this adapter.
- **BASEADA:** This is the core adapter class that provides access to any published system service. This adapter accesses system objects through the page server. Services with this adapter need to indicate the object (application service), which should be invoked.
- **STGUPADA:** This staging upload adapter class is used when an extra step is required prior to using a service with the core adapter. For example, perhaps you need to read a file, which is not in XML format, and convert it to an XML format recognized by the system. Services with this type of adapter do not need to indicate an application service but must indicate information about the file to be converted. Refer to [XAI Staging Control](#) for more information.
- **XAICMNDADA:** This is an internal adapter. It is used to send commands to the XAI Server.
- **SIEBELADA:** This adapter is no longer supported.

Designing XAI Executors

The executor is responsible for executing messages received through a message receiver. The product provides an executor, which uses the XAI server; however the architecture allows for implementing additional execution classes. If you require a different executor and therefore a different execution class, it is recommended that your implementers contact customer support.

FASTPATH: Refer to [XAI Executor](#) for more information.

Designing Message Senders

This section only describes message sender functionality that is only related to the XAI/MPL processing. Refer to [message sender](#) for details about other types of senders that are supported independent of XAI/MPL.

Message senders are responsible for define outgoing message destinations and for " [responding](#)" to the XAI executer.

- For NDS messages, the sender to use is defined on the [XAI route type](#) for the notification download profile.
- For outbound messages, the sender to use is defined on the [external system](#) / outbound message type collection.
- For responding to the XAI executer, the sender to use is defined on the [receiver](#).

For each sender, you must reference an appropriate Message Class. The information in this section describes the sender classes that are provided with the system.

You must create senders to "respond" to the various staging table receivers in the system.

- Create a sender to be used for "responses" to messages processed by the staging control receiver. You should create one sender, which points to the Message Class **UPLDERRHNDLR**. Refer to [Staging Control Sender](#) for more information about this sender.
- Create a sender to be used for "responses" to messages processed by the upload staging receiver. You should create one sender, which points to the Message Class **STGSENDER**. Refer to [Staging Upload Sender](#) for more information about this sender.
- Create a sender to be used for messages processed by the download staging receiver. You should create one sender, which points to the Message Class **DWNSTGSNDR**. (DWNSTGSNDR is not supported in all products.)
- Create a sender to be used for messages processed by the outbound message receiver. You should create one sender, which points to the Message Class **OUTMSGSNDR**. Refer to [Outbound Message Sender](#) for more information about this sender.

Next, design the senders for "responses" to other receivers, for example the JMS queue receiver or JMS topic receiver. The system provides message classes to use for these senders. Use the class **JMSENDR** for a JMS queue sender and **TPCSNDR** for a JMS topic sender.

Finally, review all your [outgoing messages](#) and determine the mechanism for communicating with the target system for each message.

- For all senders that are used for [real time messages](#), define a context entry with a context type of **Response Time Out** to define the amount of time the system should wait for a real time response.
- An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. For an HTTP sender, reference a message class of **HTTPSNDR**. In addition, the context described in [Message Sender — Context](#) for the **RTHTTPSNDR** apply to this sender as well.
- An email sender allows for XML messages to be sent as email messages through an SMTP server. It can be used in notification download processes to send a response as an email message. The email sender supports standard email functionality such as "CCs" and attachments. The content of the email message is controlled by the XSL script defined in the [XAI route type](#) of the NDS message. The XSL script has access to all context records of the NDS message as well as the input XAI message that was created by processing the NDS. Reference a message class of **EMAILSENDER**. In addition, the context described in [Message Sender — Context](#) for the **RTHTTPSNDR** apply to this sender as well.
- If you want XML messages to be written to a flat file, use a flat file sender. For example, it can be used in the notification download process to write a response message to a flat file. Flat file senders should reference a Message Class of **FLATFILESNDR**. In addition, the following context records should be defined for senders of this type.

Context Type	Description	Values
Flat file output directory	Directory in the file system where to write the file	

Context Type	Description	Values
Flat file filename pattern	The name of the output file. The file name may be a literal constant, or generated dynamically. To create a dynamic filename use <file name>\$\$ID, where \$\$ID is replaced at run time by the ID of the NDS message that triggered the response message. If no file name is defined for the sender, the XAI server generates a file name with the following format 'XAI\$\$ID.xai'.	
Append data to file	This parameter controls whether the content of the response message is appended to an existing file, or a new file is created (possibly replacing an existing one).	YES or NO
Character Encoding	Indicates if the message should be sent with character encoding. The sender will write the content of the file with encoding specified in the context value. If no value is specified, the sender uses the default Java system encoding, which is determined according to the operating system locale.	UTF-8 or UTF-16

Designing XAI Groups

XAI groups are used by the system to process an XML file containing multiple messages to be uploaded into the system. One or more groups may be defined for an [XML file receiver](#).

FASTPATH: Refer to [XML Message File](#) for more information about how groups are used to process an XML file.

When setting up your XAI environment, identify the interfaces that require uploading an XML file containing multiple XML messages into the system through XAI.

First you need to categorize the XML files that you may receive. Define an XAI Group for each logical categorization. For example, you may want to define a separate XAI Group for each third party who may send you a collection of XML messages. Or, if all third party service providers send direct access messages in a standard format, you may want to define a single XAI Group for direct access messages.

For each group, you need to identify the root elements that indicate when a new message is starting. This collection of unique root elements for a group is called the attachments.

For each group, you must identify every possible message that may be sent. For every message, define an XAI Rule. The rule indicates the XSL transformation script to be executed along with the XPath and XPath value that the system uses to identify each message.

FASTPATH: Refer to [XAI Group](#) to define groups, their attachments and their rules.

Designing XAI Receivers

Receivers define small pieces of code that wait for requests to be received through various sources. Each receiver references a Message Class where the small piece of code is defined. The following receiver classes are provided:

- **STGRCVR** The receiver that references this class polls the XAI upload staging table for new inbound requests.
- **STGCTLR**: The receiver that references this class polls the XAI staging control table for new upload processes.
- **DWNSTGRCVR**: The receiver that references this class polls the notification download staging table (NDS) for new messages. (Not available in all products).

- **OUTMSGRCVR:** The receiver that references this class polls the outbound message table for new messages.
- **JMSRCVR:** Receivers that reference this class receive requests through a message queue that supports the JMS Queue interface, such as IBM MQSeries.
- **TPCRCVR:** Receivers that reference this class receive requests through a publish/subscribe model, such as TIBCO, or any system supporting the JMS Topic interface.
- **FILESCANRCVR:** Receivers that reference this class poll a given directory for files with a given file name pattern.
- **XMLFILERCVR:** Receivers that reference this class poll a given directory for XML files with a given file name pattern.

Multiple receivers may be defined for these receiver classes. For example, the XML file receiver defines the scan directory. If you have multiple directories that contain files to be uploaded, define a receiver for each directory.

All types of receivers reference a Message Class and XAI Executer. If you require a new Message Class or Executer because you use a protocol that is not currently supported, it is recommended that your implementers contact customer support.

Designing Responses for a Receiver

Once a request has been sent for execution to the XAI server (via the executer), [the response layer](#) processes the response. For some receivers, a response may not be applicable. For example, a file scan receiver reads flat files in a given directory and posts records to the XAI staging control table. Responses are not applicable for this type of receiver.

The response may be conditional on the outcome of the request and may be sent to more than one destination (sender). To design your receiver responses, determine the conditions under which a response should be sent for each request processed by each receiver:

- Never send a response
- Send a response if the request was successful
- Send a response if the request was unsuccessful due to a system error
- Send a response if the request was unsuccessful due to an application error

Once you determine when to send a response, you must determine where to send the response. Responses for different conditions may be sent to different Message Senders or to the same Message Sender.

Designing Receivers that Poll Staging Tables

The following receivers are needed to poll the various system staging tables.

- Create a [staging upload receiver](#) that references the Message Class **STGRCVR**. For responses, **All Events** should reference the [staging upload sender](#).
- Create a [staging control receiver](#) that references the Message Class **STGCTLR**. For responses, **All Events** should reference the [staging control sender](#).
- If your implementation uses the NDS message method of communicating outgoing messages, create a staging download receiver that references the Message Class **DWNSTGRCVR**. For responses, **All Events** should reference the download staging sender. NDS messaging is not supported in all products.
- If your implementation uses the [outbound message](#) method of communicating outgoing messages, create an [outbound message receiver](#) that references the Message Class **OUTMSGRCVR**. For responses, **All Events** should reference the [outbound message sender](#).

For all the above receivers, if you need to access multiple environments, simply create receivers for each JDBC connection. Note that you should not add more than one of each of the above receivers pointing to the same JDBC connection. To improve performance for a single JDBC connection you may [configure multiple MPL servers](#).

NOTE: Sample Data for Initial Install. When first installing the system, records for each of the above receivers are provided. Your implementation may use these records or remove them and create your own.

Designing a JMS Queue Receiver

If you need to receive messages through a JMS compatible queue, you need to define a JMS Queue receiver. When designing a JMS Queue receiver you first need to design a JMS Connection and a JMS Queue.

If you would like to post [responses](#) back to the JMS queue, you may create a [message sender](#) to send the response to the JMS queue.

Designing a JMS Topic Receiver

If you need to receive messages through a JMS Topic using the publish/subscribe model, you need to define a JMS Topic receiver, which receives messages published under a specific topic. When designing a JMS Topic receiver you first need to design a JMS Connection and a JMS Topic.

If you would like to post [responses](#) through a JMS Topic using the publish/subscribe model, you may create a [message sender](#) to send the response to the JMS topic.

Designing a File Scan Receiver

The file scan receiver constantly looks in a given directory for files with a given pattern. When it finds a matching file, it creates a record in the [staging control](#) table to upload the contents of the file into the [upload staging](#) table.

When setting up your XAI environment, identify the interfaces that require uploading a file from a directory into the system through XAI. For each unique file, define a file scan receiver. For each receiver record, indicate the Scan Directory, where new files will be placed, the Scan File, which is the naming pattern to look for and the XAI Inbound Service to use for mapping the data into a system service.

In addition, if you want to specify a character encoding, the following Context record should be defined.

Context Type	Description	Values
Character Encoding	Indicates that the message is character encoded. When the receiver creates a staging control entry for a file, it will add ?enc=?x to the name of the file in the table where x is the value of this parameter. Refer to Sequential Input File for more information.	UTF-8 or UTF-16

Designing an XML File Receiver

The XML file receiver constantly looks in a given directory for XML files with a given pattern. When it finds a matching file, it goes through steps to identify each separate message in the file, determine the appropriate XSL transformation and create a record in the staging upload table.

FASTPATH: Refer to [XML Message File](#) for more information.

When setting up your XAI environment, identify the interfaces, which require uploading an XML file containing multiple XML messages into the system through XAI. For each unique file, define an XML file receiver. For each receiver record, indicate the Scan Directory, where new files will be placed, the Scan File, which is the naming pattern to look for and the collection of XAI groups. XAI groups are used by the system to identify each separate message in the file and to determine the appropriate XSL transformation for each message.

FASTPATH: Refer to [Designing XAI Groups](#) for more information.

Configuring the System for NDS Messages

Refer to the documentation for your product to find out if NDS messaging is supported.

Schema Editor

CAUTION: This schema editor is a separate legacy tool that is no longer recommended. The documentation remains for upgrade purposes

The Schema Editor is a Graphical User Interface (GUI) tool to create XML schemas. The tool provides wizards to generate schemas from various sources.

Opening the Schema Editor

After launching the schema editor, you are asked to connect to a database. On the Connect dialog:

- Select an ODBC data source pointing to the desired database.
- Enter the data source, user ID, password and database owner required to log on to that database.
- Click **Connect**.

After connecting, the schema editor appears.

Use the File/Open dialogue to select a schema from the schema directory. Refer to [The Options Menu](#) for information about setting the default schema directory.

NOTE: When opening a schema, the schema editor validates the schema and any errors are displayed. Refer to [Validating a schema](#) for more information.

Schema Editor Window

CAUTION: This schema editor is a separate legacy tool that is no longer recommended. The documentation remains for upgrade purposes

The schema editor allows you to modify individual elements and attributes of a given schema.

Description of Page

Refer to [System Wide Functions for Schema Editor](#) for information about the various menu options available for the schema editor.

Service Name Enter the name of the service to be created in the service name text box. This is the name of the first element under the Body element in the XML document.

CAUTION: Important! When adding new schemas, carefully consider the naming convention for the Service Name. Refer to [System Data Naming Convention](#) for more information.

Adapter The adapter used to process services using this schema.

Internal Service Name If the schema is for an adapter that should invoke a system service, this is the internal name of the service.

Transaction Type Select the transaction type performed by the service. The available values are **Read, Add, Change, Update, Delete, List** and **Search**.

NOTE: The difference between Change and Update is that for Change, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For Update, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

Left Panel

The left panel of the schema editor displays a tree view of the hierarchical elements in the schema. The (+) expands a node, the (-) collapses a node.

Right Panel

The following attributes appear on the right panel of the Schema Editor. Some fields cannot be modified in the schema editor. The field description indicates when the field is protected.

Tag Name The XML element tag name. This field is protected, but you may modify this attribute to give the element a self-explanatory name by right-clicking on the element name in the left tree-view.

MetaInfo Name Maps the element to a fully qualified field name in the service, for example PER_ID. This field is protected.

Internal Type This property is populated automatically when you generate the schema from your product. The values further define elements and attributes. The values are **page**, **pageBody**, **list**, **listHeader**, **listBody**, **searchHeader**, **codeTableDesc**, **Private**. The values of **codeTableDesc** and **Private** are used to define special types of attributes.

Private attribute A field that does not exist on the server side, but one that you still want to have in the schema.

Description A description of this field.

Content The element type. This field is only available for elements. Possible values are **eltOnly**- element may contain only other elements and no text, **TextOnly**- element may only contain text.

Search Type Services, which perform a Search, may allow searching based on different criteria. The values are taken from the system meta information when the schema is generated. The possible values are **Main**, **Alternate1**, **Alternate2**, **Alternate3**, **Alternate4**, **Alternate5** and **Alternate6**.

NOTE: You would not typically modify this value because it corresponds to a value in the meta information. However, the value is modifiable to accommodate the rare case in which a service may change in a new release. In this scenario, you may prefer to update the schema manually rather than regenerate a new schema for the new version.

Is Part of Primary Key Used to indicate to the XAI server whether or not this field makes up part of the primary key of the record. The values are taken from the metadata information when the schema is generated. Value may be **true** or **false**.

NOTE: Typically you would not modify this value because it corresponds to a value in the meta data information. However, the value is modifiable to accommodate the rare case where a service may change in a new release. In this scenario, you may prefer to update the schema manually rather than regenerate a new schema for the new version.

Min Occurs This field is available for elements only and is used for repeating elements. It defines the minimum number of occurrences for an element. Value may be 0 or 1.

Schema Max Occurs This field is available for elements only and is used for repeating elements. It defines the maximum number of occurrences for an element. Value may be 0, 1 or *.

Limit Number of occurrences This field is available for elements only and is used for repeating elements. If the **Schema Max Occurs** field has been set to '*', define the number of max occurrences here.

XML Data Type The data type for the attribute. Possible values are **number**, **string**, **decimal**, **date**, **dateTime**, and **boolean**.

Server Data Type Indicates the data type of this attribute on the server. This field is protected.

Service Format The format expected by the service. At runtime, XAI converts the Tag format to the Service Format before executing the request. Formats are defined in [XAI Format](#).

Tag Format The format used to format an element/attribute in the schemas. Formats are defined in [XAI Format](#).

Min Length Use this property to define the minimum length of the attribute, if applicable.

Max Length Use this property to define the maximum length of the attribute, if applicable.

Precision This is used for decimal attributes to define the maximum number of digits.

Scale This field is used for decimal attributes to define the number of digits at the right of the decimal point.

Required A value of **Y** indicates that the element must appear in XML document. A value of **N** indicates that the element is optional.

Default value Default value to be used for Request schema, when the element is not supplied as part of the XML request document.

Fixed Value Fixed value to be used for Request schema. This value is used regardless of the value supplied in the request document.

Code Table Field This property is used for attributes that are descriptions of a code table, where the description is not automatically returned by the system service. Use this property to indicate the code whose description should be retrieved by the XAI server.

Code Table Program This property is used for attributes that are descriptions of a code table, where the description is not automatically returned by the system service. Use this field to indicate the program that XAI should call to access the description for the **Code Table Field**.

Creating a Schema

Usually you do not create schemas from scratch; rather you use Schema Creation Wizards to import existing data structure definitions from a variety of data sources:

- System services
- Comma Delimited Files
- Database Extract
- Any XML document

Once a schema is created based on the existing data structure, it is displayed in a TreeView on the left panel. Once the imported schema has been edited, it serves as the basis for creating the request and response schemas. When imported, the schema exposes all fields defined in the service. You may want to remove some attributes/elements from the request or response schema.

NOTE: Although the main purpose of the editing process in the creation of the request and response schemas is the elimination of elements, which makes the schema shorter and more understandable, it is not required for processing purposes. Therefore, if you don't mind that you have not used elements in your schemas, you could stay with one schema, which serves as both the request and response schema.

1. Save the Schema as a Request schema with an appropriate name, for example PersonInfoRequestSchema.xml
2. To create the Response schema, which is identical to the request schema, use the Save As Response menu option. This renames the top element of the schema to ServiceNameResponse, for example PersonInfoResponse and save the schema under a different name i.e. PersonInfoResponseSchema.xml. Note that if the request and response schemas are identical then one schema may be used for both and there is no need to create separate schemas.
3. Read in the Request Schema (File/Open) and modify its structure. Depending on the service type, you'll have to modify the contents of the Request Schema. This is usually required when the service is an "Info" service, which requires very few input elements. In such cases you'll delete most of the elements on the schema and only leave the necessary elements required to invoke the service. For example: in the PersonInfo request, you only need the PersonId and the Company elements in the request schema.
4. Read in the Response Schema (File/Open) and Modify its structure. Depending on the service type, you'll want to modify the contents of the Response Schema. This is usually required when the service is an "Add" or "Delete" service, which returns very few input elements. In such cases you'll delete most of the elements on the response schema and only leaves the necessary elements required by the requester of the service.

Adding an Element/Attribute

Usually, you won't have to add element or attributes to a schema. However if the schema already exists and you want to add an element/attribute, you can follow this procedure. Be aware that any element/attribute added here must also exist on the xml metainfo.

- Select the element's node on the TreeView.
- Right click on it and select the 'Add Element' or 'Add Attribute' option in the pop-up menu
- Enter the element/attribute name in the prompt dialog box and click OK.

Removing Elements/Attributes

When generating a schema using one of the wizards, the generated schema may contain information that you do not want to publish as part of the service, or is not required for a particular service. You can remove elements/attributes from the schema, and though these elements/attributes may still exist on the service they are not seen by the XAI service using this schema. To remove an element or attribute:

- Select the node to be removed.
- Right click and select "Delete" from the popup menu.

Renaming an Element


To rename an element:

- Select the element's node on the TreeView.
- Right click it and select the Rename option in the pop-up menu
- Enter the new name in the prompt dialog box and click OK.

NOTE: The information in this table is cached by the XAI server and by the MPL server. Changes to values in this table do not affect the runtime copy of these servers. Refer to [XAI Command](#) for information about refreshing a schema.

Validating a Schema

CAUTION: This schema validation is part of a separate legacy tool that is no longer recommended. The documentation remains for upgrade purposes

Although a schema is validated against the metainfo XML file when it is read into the editor or before it is saved, you can perform the validation at any time while the schema is being edited. To validate a schema, click on the toolbar "Validate" button . If the schema fails to validate the schema errors dialog is displayed.

When the editor fails to validate a schema against the xml metainfo file, it pops up a dialog that lists the errors found in the schema definition. These errors may be of two types:

- An element or attribute in the schema could not be found in the xml metainfo file. You can click **Remove** to remove the element from the schema.
- The data type of an attribute does not match the one defined in the metainfo file. You can click **Correct**, and the editor fixes the data type so it does match.

The **Correct All** button can be used to correct all fields that have data types that do not match the one in the XML metainfo file.

NOTE: Correcting errors does not save the schema definition into a file. You have to save it manually.

If you **Exit** without correcting the errors, the schema displays with the mismatch information highlighted in red.

Registering a Service

Before a service can be used it must be defined in the [XAI Inbound Service](#) table in the XAI registry. A service can be registered in the XAI registry directly from the schema editor. Go to the menu item 'Register Service' in the 'Schemas' menu. The Register service UI page appears. Fill in the required fields.

NOTE: The registry entry for the service can be later modified using the [XAI Inbound Service](#) page.

Testing a Schema

The XAI schema editor provides a testing option.

NOTE: Testing in your product. You may also test your schemas and your services using [XAI Submission](#).

System Wide Functions for Schema Editor

Because the schema editor is in an application outside of the standard products, this section introduces some general functions related to the application.

Application Standards

- The schema editor is a Multiple Document Interface (MDI) windows application. It may contain multiple active windows. You may jump from one window to the other using the Window menu option.
- The Editor window is always active. Closing the schema editor window quits the application.
- In the Editor window, a splitter bar is available to resize the schema tree view horizontally.
- Actions may be performed using menu items or by clicking on toolbar buttons.
- All messages are displayed on a status bar at the bottom of the screen. Some may also be displayed using message boxes.

The File Menu

Use the File menu to open existing schemas and to save a schema to a file.

Connect — Connects to the database.

Open — Loading an existing schema into the editor

You can read an existing schema into the editor.

1. Click on the Open toolbar button or select the File/Open menu option.
2. A file selection dialog is shown. Select the schema file name.
3. The editor first validates the schema against the xml metainfo file. If it fails to validate it shows the Schema Validation Errors dialog. Refer to [Validating a schema](#) for more information.

Save — Save the current schema to a file. Using the current file name.

To save a Schema, click on the Save toolbar button, or select the File/Save menu option. When you save a schema, the editor first attempts to validate the schema. If it fails to validate it against the XML Metainfo file, you are prompted to save it with inconsistency errors or to return to the editor.

Save As — Save the current schema to a file. Use a different file name.

Save As Response Save a copy of the current schema to a file as a response schema. Use a different file name.

The View Menu

Use the view menu to perform actions on the Tree View nodes or to view errors. The following menu options are available:

Expand All — Expand all nodes in the Tree View

Collapse All — Collapse all nodes in the Tree View

Expand Branch — Expand the selected node and all the node's children

Search (Ctrl+F) — Find a node with a node name containing a given string

Search Again (F3) — Find the next node with containing the search string

View Schema Errors — Display the Schema Validation Errors dialog.

Web Browser — Display the current schema definition on a web browser page

The Schemas Menu

Use the Schemas menu to create, test, validate and register schemas.

- To create schemas from various sources use the **Create menu**.
- To [validate a schema](#) select the **Validate** option (Ctrl+V).
- To create a sample instance and test the schema, select the **Test** option (Ctrl+T).
- To create an entry in the service registry for a service represented by the current schema, select the **Register Service** option (Ctrl+R). Refer to [Registering a Service](#) for more information.

The Options Menu

Always Save As W3C — Turn on this option to save schemas in W3C format by default instead of XDR format.

Always Save As DTD — Turn on this option to save schemas in DTD format by default instead of XDR format.

Preferences

The following options may be set on the preferences dialog (Select Options and then Preferences):

- The Default Date Time Format - used for schema date fields
- The default Schema Directory - used to read/save schemas
- The default Metainfo Directory - used to create/validate schemas
- The XAI Servlet URL - used when executing requests from the test schema dialog.
- The Default Account Id - used when generating sample instances of a schema
- Language - used to access language specific data

The Tools Menu

Schemas tools can be invoked from the **Schema Editor Tools** menu.

Converting Schemas to a W3C compatible schema — Schemas generated in the Microsoft BizTalk-compatible format (XDR format) may be saved in a format compatible with the *October 2000, W3C XML* schema standard. To save a schema in a W3C format:

- Select **Convert to W3C** from the **Tools** menu. The **Convert to W3C** dialog box appears.
- Select the schema(s) to be converted. Multiple schemas may be converted in a single step.
- The name of the W3C schema is the same name as the schema but with an ".xsd" file extension, and is saved to the same directory as the original schema.
- Click **Convert**.

Converting Schemas to a DTD — Schemas generated in the Microsoft BizTalk-compatible format (XDR format) may be saved as a DTD.

- Select the **Convert to DTD** option from the **Tools** menu. The **convert DTD** dialog box appears.
- Select the schema(s) to be converted. Note that multiple schemas may be converted in a single step.

- The name of the W3C schema is the same name as the schema but with a ".dtd" file extension, and is saved to the same directory as the original schema.
- Click **Convert**.

Validating multiple schemas — The schema validation tool can be used to validate the correctness of an XAI schema when compared to the metainfo xml definition used to generate the schema. For each validated schema, the validation tool scans the list of elements/attributes and compares them with those defined in the XML metainfo file. Select **Validate Schemas** in the **Tools** menu.

- The **Validate Schema** dialog box appears.
- The left list box is used to select the directory where the schemas to be validated are stored. By default this is the **Schema Directory** as defined in the preferences.
- On the right, the list of schemas is displayed on a grid.
- Multiple schemas may be validated in a single step.
- To select a schema click on the left button (the first column in the row).
- To select multiple schemas, hold the `Ctrl` key and select the required schemas.
- Click `Validate Schemas`.
- The schema grid is updated. When an attribute defined in the schema is missing from the metainfo file or when the properties of the element do not match defined in the metainfo, a button is displayed at the right of the schema name. Clicking on the **edit** button loads the schema into the editor and displays the Schema Validation Errors dialog for that schema. You can correct the schema and save it. Refer to [Validating a schema](#) for more information.

Setting Up Your XAI Environment

This section describes the control tables available to administer your XAI environment.

Message Class

The Message Classes are references to actual Java classes. The Message Class defines the Java class used to implement Receivers, Senders, Adapters and Executors. This information is provided with the system and does not need to be modified for a specific installation.

To view a Message Class, open **Admin > XAI > Message Class**.

Description of Page

The **Message Class** and **Description** are unique identifiers of the Message Class. The **Class Definition** indicates the Java class, implementing the adapter, receiver, sender or executor.

Owner indicates if this Message Class is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a Message Class. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_CLASS](#).

XAI Envelope Handler

To view your envelope handlers, open **Admin > XAI > XAI Envelope Handler**. This information is provided with the system and does not need to be modified for a specific installation.

Description of Page

Enter a unique **XAI Envelope Handler ID** and **Description**.

Indicate whether the **Envelope Type** is **Custom SOAP Envelope**, **Default** (no SOAP environment) or **SOAP Envelope**. Note that the values of **Siebel Integration Message** and **Siebel VBC** are not supported.

When the envelope type is **SOAP Envelope**, indicate the **Envelope URI**.

Owner indicates if this XAI envelope handler is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI envelope handler. This information is display-only.

Setting Up Your Registry

The following section describes the control tables that are logically considered part of the XAI Registry.

XAI JDBC Connection

To view an XAI JDBC Connection, open **Admin > XAI > XAI JDBC Connection**.

Description of Page

Enter a unique **XAI JDBC Connection** and **Description**.

Use the **Connection Type** to indicate how the JDBC connects to a database. The following connection types are valid:

- **Oracle Defined Connection** indicates the connection is to an Oracle database through a JNDI entry.
- **DB2 Defined Connection** indicates the connection is to a DB2 database through a JNDI entry.
- **JNDI Defined Connection** indicates the connection is using the MQ series classes implementing JMS.
- **Determined by parameter file** indicates that the connection information should be determined by looking at the parameters defined at [Installation](#).

For connection types of **Oracle** or **DB2**, use the **JDBC URL** to indicate URL of the database connection to be initialized at XAI/MPL startup time. Indicate the **Database User** and **Database Password** required for accessing the database. The JDBC connection URL can either be a Type 2 or a Type 4. For example:

- Type 2: jdbc:oracle:oci8:@CD200ODV
- Type 4: jdbc:oracle:thin:@myhost:1521/ CD200ODV

For a connection type of **Determined by parameter file**, indicate the parameter substitutions, which should be accessed from the parameter file for the JDBC URL, database user and database password, for example, @JDBCURL@, @DBUSER@ and @DBENCPASS@.

When the connection type is **JNDI**, indicate the **JNDI Server** and the **JNDI Data Source** name as defined in the JNDI.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_JDBC_CON](#).

XAI Format

Open **Admin > XAI > XAI Format** to define the various formats.

Description of Page

For each new format, specify a unique **XAI Format** name and **Description**.

Indicate whether the Format Type is a **Currency formatting string**, a **Date/Time formatting string**, a **Phone formatting string** or a **Text formatting string**.

Finally, indicate the **Format Expression**, which defines the formatting pattern to be applied.

Owner indicates if this format is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI format. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_FORMAT](#).

XAI Adapter

To define a new adapter, open **Admin > XAI > XAI Adapter**.

Description of Page

Indicate a unique **Adapter Name** and **Description**.

Indicate the **Message Class**, which is the name of the Java class, implementing the adapter. The class should be one that is defined for an adapter. The adapter classes provided with the product are **BASEADA**- Core Adapter, **BUSINESSADA**- Business Requests Adapter and **XAICMNDADA**- XAI Command Adapter. Note that the values **SIEBELADA** and **STGUPADA** are no longer supported.

FASTPATH: Refer to [Message Class](#) for more information.

The following fields are not applicable for the **BusinessAdapter** adapter.

Use the **JNDI Server** to indicate the name of the WebLogic JNDI server running your product. Refer to [JNDI Server](#) for more information.

Indicate the **Default User** to be passed to your product server when this adapter is executed.

NOTE: If the XML request is sent over an HTTP connection, which has been authenticated, the authenticated User Id is passed to your product.

The **Default Date** format and the **Default DTTM** (date / time) **Format** specify date and date/time [formats](#) to use when a schema does not explicitly indicate formats.

Owner indicates if this XAI adapter is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI adapter. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_ADAPTER](#).

XAI Executer

To define a new Executer, open **Admin > XAI > XAI Executer**.

Description of Page

Enter a unique **Executer ID** and **Description**.

Indicate the **Message Class** for this executer. The class should be one that is defined for an executer. The executer class provided with the product is **XAIURLEXEC**- XAI Executer.

Indicate the appropriate **Executer URL**.

Owner indicates if this XAI executer is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI executer. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_EXECUTER](#).

XAI Group

XAI groups are used to process XML files, which contain a collection of [XML messages](#) to be uploaded in batch.

XAI Group - Main

To define your XAI groups, open **Admin > XAI > XAI Group**.

Description of Page

Enter a unique **Group** and **Description** for the XAI Group.

Indicate the **Parser** used for this group. Possible values are **Dom Parser** and **StAX Parser**.

NOTE: **Dom Parser** reads the full XML document into memory and therefore is not ideal for larger XML documents.

Indicate the **XPath** and **XPath Value**, which an XML file receiver uses to identify which group a given XML file belongs to.

- For **StAX Parsers** the XPath is limited to the root element.
- For **Dom Parsers**, the XPath supports defining elements at a lower level than the root element.

XAI Group - Attachments

Open **Admin > XAI > XAI Group** and navigate to the **Attachments** tab to define attachments for your group.

Description of Page

For each entry in the attachments collection, indicate the **Sequence** and the **Root Element**. Use **Include Elements** to indicate if **Parent** elements should be included along with the current element when applying the XAI rules.

FASTPATH: Refer to [XML Message File](#) for more information about how this is used.

XAI Group - Rules

Open **Admin > XAI > XAI Group** and navigate to the **Rules** tab to define rules for your group.

Description of Page

For each entry in the rules collection, indicate the **Sequence**, the **Priority**, the **XPath** name and **XPath Value** and the **XSL File Name**.

NOTE: **Include Parent**. If your attachment indicates that **Parent** elements should be included, be sure that the parent element is included in the XPath defined here.

FASTPATH: Refer to [XML Message File](#) for more information about how this is used.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_RGRP](#).

XAI Receivers

XAI Receiver - Main

To define your XAI receivers, open **Admin > XAI > XAI Receiver**.

Description of Page

Enter a unique **Receiver ID** and **Description** for the XAI Receiver.

Indicate the **Message Class** for this receiver. The class should be one that is defined for a receiver. The receiver classes are **DWNSTGRCVR**- Download Staging receiver, **FILESCANRCVR**- Upload Files from a directory, **JMSRCVR**- JMS Queue receiver, **OUTMSGRCVR**- Outbound Message receiver, **STGCTLRRCVR**- Staging Control receiver, **STGRCVR**- Staging Upload Receiver and **TPCRCVR**- JMS Topic receiver, **XMLFILERCVR**- XML File receiver.

FASTPATH: For more information, refer to [Designing XAI Receivers](#) about different types of receivers.

Indicate whether or not this receiver is currently **Active**.

Identify the **Executor ID**. Select the XAILOCAL executor if the Message Class for this receiver is STGCTLRVCVR. Select the BYPASSXAI executor if the Message Class for this receiver is OUTMSGRCVR. For all other receivers select the XAIURL executor. For more information, refer to [XAI Executor](#).

Indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**.

The **Read Interval** indicates the number of seconds between read cycles.

Start At Time and **Duration** are not currently in use.

If the Message Class for this receiver is **FILESCANRCVR**, **STGRCVR**, **STGCTLRVCVR** or **XMLFILERCVR**, indicate the **XAI JDBC Connection**.

FASTPATH: Refer to [XAI JDBC Connection](#) for more information.

Turn on **Sequential Execution** if the received requests should be processed in [sequential order](#) (instead of multithreaded). If this value is turned on then XAI staging control records created by this receiver are marked for sequential execution.

JMS Information

The following information is only available if the Message Class is **JMSRCVR** or **TPCRCVR**.

Indicate the appropriate **JMS Connection**

FASTPATH: Refer to [JMS Connection](#) for more information.

Indicate the appropriate **JMS Queue**.

FASTPATH: Refer to [JMS Queue](#) for more information.

Indicate the appropriate and **JMS Topic**.

FASTPATH: Refer to [JMS Topic](#) for more information.

File Information

The following information is only available if the Message Class is **FILESCANRCVR** or **XMLFILERCVR**.

Use the **Scan Directory** to indicate where to look for new files.

In **Scan File**, indicate the file pattern. All files with names matching the pattern are uploaded into the staging upload table. For each file found, a record in the staging control table is created.

CAUTION: WARNING. MPL expects all files conforming to the Scan File pattern to be complete. If a file is in the process of being copied into the scan directory and its name conforms to the naming pattern, MPL still attempts to process it and may issue an error for the incomplete file. It is suggested that files first be copied into the scan directory with a different name that does not conform to the naming pattern, for example filename.xml.inprocess. Once the file copy/transfer is complete, rename the file to one that conforms to the naming pattern, for example, filename.xml.

The following information is only available if the Message Class is **FILESCANRCVR**.

Use the **XAI In Service Name** to indicate how the records in the file are mapped and how they are transformed to match a system service request structure.

XAI Receiver - Context

Open **Admin > XAI > XAI Receiver** and navigate to the **Context** tab to define context for your receiver.

Description of Page

The Context collection enables you to define a collection of **Context Types** and **Context Values** defining. Use this collection when you need to store an attribute of a receiver that is not catered for in the current table.

NOTE: The values for the Context Type field are customizable using the Lookup table. This field name is RCVR_CTXT_FLG.

XAI Receiver - Response

Open **Admin > XAI > XAI Receiver** and navigate to the **Response** tab to define where to send responses to requests made by this receiver. Refer to [Designing Responses for a Receiver](#) for more information.

Description of Page

The response collection enables you to define the destination (**Message Sender**) where responses to a request may be sent under various circumstances (**Event**). The events currently defined with the product are **All events, Message executed OK, Application Error, System Error**.

NOTE: The values for this field are customizable using the Lookup table. This field name is ON_EVENT_FLG.

XAI Receiver - Groups

Open **Admin > XAI > XAI Receiver** and navigate to the **Groups** tab to the valid XAI groups for an XML file receiver.

Description of Page

This collection is only available if the Message Class is **XMLFILERCVR**.

For each entry in the Group collection, indicate the **Priority** and the **Group**. Refer to [XAI Groups](#) for more information about defining groups.

Where Used

Receivers are used by the XAI server and by the MPL server to process messages sent to the system from various sources.

XAI Inbound Services

The XAI Inbound Services section in the registry is the main section of the registry. It is used to define the service characteristics. Basically, a service is defined by an Adapter responsible for executing the service, a pair of XML schemas and connection attributes. The Adapter defines the interface with the target application server, while the schemas define the structure of the request XML document expected by the service and the structure of the response XML document generated by the service.

XAI Inbound Service - Main

To update an inbound service, open **Admin > XAI > XAI Inbound Service**.

CAUTION: Important! When adding new inbound services, carefully consider the naming convention of the XAI In Service Name. Refer to [System Data Naming Convention](#) for more information.

Description of Page

Define a unique **XAI In Service Name**. This information is used in the system to identify the service. The service name is also the first XML element after the <Body> element in the XML request/response document. The system generates a unique **XAI Service ID**, which serves as the primary key.

Owner indicates if this XAI inbound service is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI inbound service. This information is display-only.

Indicate the **Adapter**, which defines the interface with the target application server.

FASTPATH: Refer to [XAI Adapter](#) for more information.

If adapter for this service should invoke a system service, then indicate the appropriate **Service Name**.

FASTPATH: Refer to [Service Program](#) for more information about defining services.

If adapter is the base package **Business Adapter** then **Service Name** does not appear. Instead, use **Schema Type** to indicate the type of object this service invokes and **Schema Name** to reference the object to invoke. Using this adapter, you may set up service to invoke [business objects](#), [business services](#) and [service scripts](#).

FASTPATH: Refer to [Designing XAI Adapters](#) for more information about the **Business Adapter**.

Use the **Description** and **Long Description** to describe the service.

Check the **Active** switch if this service is enabled and available for execution. If this switch is unchecked, then the service is defined in the registry, but not yet available for public use.

Check the **Post Error** switch to support [inbound message error handling](#) for messages that are not processed via the staging upload table.

Check the **Trace** switch if you would like the trace to be on for this particular service. If the general trace option is not activated, you can force a trace for a particular service.

FASTPATH: Refer to [Server Trace](#) for more information about trace functionality.

When the **Debug** switch is checked, debug information is generated on the XAI console when this service is executed. The debug information can be useful to resolve problems.

Schema Definitions

NOTE: Request Schema and Response Schema are not applicable to services invoking schema-based objects. They do not appear when the **Business Adapter** is used.

The next two properties define the request and response XML schemas. The schemas were created using the [Schema Editor](#) and are SOAP compatible. The schema XML files are expected to be stored in the Schemas Directory on the Web server running the XAI server.

The **Request Schema** is the XML schema defining the service request. The request sent to the server must adhere to the schema definition.

The **Response Schema** is the XML schema defining the service response. The response generated by the XAI server corresponds to the response schema definition.

The same service may perform several actions on a business object. Use the **Transaction Type** to define the default action performed by a service. The transaction type can be provided when invoking a service, by dynamically specifying a transaction type attribute on the Service element of the XML request. This field may take the following values: **Read, Add, Change, Update, Delete, List** and **Search**.

NOTE: The difference between **Change** and **Update** is that for **Change**, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For **Update**, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

Services, which perform a Search, may allow searching based on different criteria. When the Transaction Type value is **Search**, use the **Search Type** to define the default search criteria. The possible values are **Main, Alternate1, Alternate2, Alternate3, Alternate4, Alternate5** and **Alternate6**.

NOTE: This is a default definition only and it may be overridden at run time when the service is invoked. To override the search type at run time, you should specify the searchType attribute on the Service element of the XML request.

XSL Transformation Definitions

Sometimes, the XML request document does not conform to the request schema, or the response document expected by the service requestor is not the one generated by the adapter. In such cases the request and/or the response documents must be transformed. The XAI server supports transformation through XSL transformation scripts. Transformation scripts may be applied to the request before it is passed to the adapter or applied to the response document before it is sent to the service requestor.

The **Request XSL** is the name of the XSL transformation to be applied to the request document before processing it. The transformation is usually required when the incoming document does not correspond to the XAI service request schema therefore it has to be transformed before it can be processed by the adapter.

The **Response XSL** is the name of the XSL transformation to be applied to the response document when the requester of the service expects the response to have a different XML document structure than the one defined by the response schema for the service.

Click the **WSDL URL** hyperlink to launch a separate window that contains the WSDL definition for the inbound service. Note that the server name and port number for the URL are built using a setting in the common properties file using the XAI HTTP Caller URL setting.

NOTE: Refer to [WSDL Catalog](#) for information on how to obtain the WSDL catalog for all XAI Inbound Services.

XAI Inbound Service - Staging

The staging tab is used to define parameters for services that use the Staging Upload adapter.

FASTPATH: Refer to [XAI Upload Staging](#) for more information.

Open **Admin > XAI > XAI Inbound Service** and navigate to the **Staging** page to define attributes for your upload staging adapters.

Description of Page

Indicate the **Staging File Type** to be processed by the staging upload service. Possible values are **Comma Delimited file**, **Database Extract** and **Sequential file**.

The format of the records in the input file are not in an XML format and do not correspond to an XAI service schema. As a result, the input record must be transformed into an XML message that conforms to an XAI service request schema. Enter the **Record XSL**, which indicates the XSL transformation script used to transform the input record into the appropriate XML message.

For **sequential files** and **Comma delimited files**, indicate the **Input File Name** to be processed.

NOTE: This parameter can be overridden in the **Staging Control** table when a request to execute such a service is made.

When the service takes its input from a **Database extract**, indicate the **JDBC Connection** used to connect to the database that contains the input data.

NOTE: If this value is not populated XAI uses the default JDBC connection, which is the current product database.

FASTPATH: Refer to [XAI JDBC Connection](#) for more information about defining these values.

Use the **Interface Name** to provide a description of the interface being implemented through this service.

XAI Inbound Service - Parameters

This tab enables you to define parameters that are used as selection criteria by the DB Extract staging upload service.

Open **Admin > XAI > XAI Inbound Service** and navigate to the **Parameters** page.

Description of Page

The **Parameters** that were defined under the Request element in the schema are displayed here. They are used to drive the extraction process. This tab only displays the list of parameters. The values for these parameters can later be entered when the control record to invoke this service is created.

FASTPATH: Refer to [Staging Control Parameters](#) for more information.

Owner indicates if this XAI inbound service is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI inbound service. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_XAI_IN_SVC](#).

NOTE: The information in this table is cached by the XAI server and by the MPL server. Changes to values in this table do not affect the runtime copy of these servers. Refer to [XAI Command](#) for information about refreshing a service.

XAI Route Type

Refer to the documentation for your product to find out if XAI Route Type is supported.

Running Your XAI Environment

NOTE: The XAI functionality is legacy functionality and not recommended for new implementations. This information remains in the documentation for upgrade purposes. Refer to [Incoming Messages](#) and [Outgoing Messages](#) for information about the recommended features for supporting sending and receiving external messages.

The [XML Application Integration](#) (XAI) module provides the tools and infrastructure that businesses require for integrating their third-party systems with the application.

This section describes the pages used to manage incoming and outgoing information via the XAI tool.

XAI Staging Control

Refer to [Staging Control](#) for more information about the purpose and functionality of this page.

Navigate to this page using **Menu > Integration > XAI Staging Control**.

Description of Page

The **XAI Staging Control ID** is a system generated unique identifier of this record.

The **XAI Staging Control Status** indicates the current state of this record.

Pending This status indicates that this record needs to be processed.

In Progress This status indicates that the background process, which uploads the staging records, is currently processing this record.

Error This status indicates that the background process found a problem with this record. When the data related to this record has been fixed, change the status back to **Pending** so that it will be processed again.

Complete This status indicates that this record has been processed.

The **Number of Uploaded Records** indicates how many XAI staging records related to this staging control record were uploaded.

The **Run Date Time** indicates the date and time this record was processed by the upload process.

The **User**, who created this staging control record, is captured.

If the data to be uploaded is found in a file, indicate the **Upload File Name**, which should include the directory path and file name.

NOTE: Character Encoding. Refer to [Sequential Input File](#) for information about specifying character encoding for a file.

NOTE: Complete File. MPL expects this file to be complete. If the file is in the process of being copied into the directory, MPL still attempts to process it and may issue an error for the incomplete file. The staging control record should only be created once the complete file is ready for upload.

Enter a description of the **XAI Interface Name**. This is simply information and is not used by the system.

Turn on **Sequential Execution** if you want the XAI upload staging records related to the staging control record to be processed in [sequential order](#) (instead of multithreaded).

The **XAI Service ID** tells the system how to create the XML related to the XAI upload staging record to be created. In Oracle Utilities Customer Care and Billing, for example, XAI Services with an adapter type of **CISStagingUpload** may be indicated here.

FASTPATH: For more information about XAI services, refer to [XAI Inbound Service](#).

Use the **Comments** to provide free format information related to this staging control record.

The collection of staging control parameters is used to define selection criteria when the staging control is accessing data from a database table. The **Staging Control Parameter** defines the field used in the WHERE clause and **XAI Staging Control Parm Value** defines the value to be used in the WHERE clause.

Refer to [Staging Control Parameters](#) for more information.

The **Message** collection displays any completion messages for XML requests related to the staging control record.

XAI Upload Staging

The XAI upload page allows you to view the details or to fix errors for a request in the Staging Upload table. This page displays the text of the XML request document linked to the upload.

XAI Upload Staging - Main

To view or modify an XAI upload document, navigate to **Menu > Integration > XAI Upload Staging** and navigate to the main tab.

Description of Page

The **XAI Upload Staging ID** is a system assigned identifier of the XAI upload staging record.

The **Application Service** identifies the internal system service called by the XAI tool to load/update the system data.

NOTE: The upload process does not use this. Rather, the application service is part of the XML Request. This field is used to help in searching for XAI Upload records.

The **XAI Staging Control ID** related to this XAI upload staging record is displayed.

The status of the upload is indicated by the **XAI Upload Staging Status**. Values are **Pending**, **In Progress**, **Complete**, and **Error**. If the record is in error, an error is written to the [XAI Upload Exception](#) table.

Create Date/Time indicates when the record was posted.

Completion Date/Time indicates when the work was completed.

If this XAI upload staging record is a response to an **XAI Download Staging** record, information about the related download staging record is displayed.

If there is an error with this record, you will see the error **Message** associated with this record. The message area is suppressed if there are no problems with the record.

Click the adjacent button to view the long explanation. The long explanation provides information about the cause of the error (and how to fix it).

The upload staging data appears in the **XML Request** text box.

XAI Upload Staging - Response

To view the response to a completed XAI upload staging document, navigate to **Menu > Integration > XAI Upload Staging** and go to the **Response** page.

Description of Page

The **XAI Upload Staging ID** is a system assigned identifier of the XAI upload staging record.

The system's response to the upload XML appears in the **XML Response** text box.

Uploading XAI Staging Records

Staging Control Layout

In order to load XML requests from flat sequential files, comma separated value (CSV) files and from database tables, you need to create a staging control record. The name of this table is **CI_XML_STG_CTL**. The following table describes each column on this table.

Column Name	Length	Req'd	Data Type	Comments
XML_STG_CTL_ID	10	Y	N	This is the unique identifier of the record. This value does NOT have to be a random number, but it does need to be unique.
RUN_DTTM	26	N	Date/Time	Date and time this record was executed.
XAI_IN_SVC_ID	10	Y	A/N	This is the XAI service associated with this record. This service should have an adapter of CISStagingUpload .
XML_STG_FILE_NAME	250	N	A/N	This is the location and name of the file containing the data to be uploaded. This is used for comma delimited files and flat sequential files.
XML_INTFC_NAME	30	N	A/N	The name of the interface used to populate this table. This is simply information and is not used by the system.
XML_STG_STATUS_FLG	2	Y	A/N	Must be set to P (Pending) .
NT_XID_CD	30	N	A/N	This field is not in use.
NT_UP_XTYPE_CD	30	N	A/N	This field is not in use.

USER_ID	8	Y	A/N	The ID of the user who created this record.
COMMENTS	254	N	A/N	Use the free format comments, if necessary.
NBR_RECORD_UPLD	10	N	A/N	Leave this blank. This will be populated by the process which creates upload staging records for this control record.

Staging Control Parameters Layout

If your staging control record should read a database table, the XAI service may include selection criteria in its WHERE clause. If that is the case, then your staging control record should populate the selection criteria. The name of this table is **CI_XML_STGCTL_P**. The following table describes each column on this table.

Column Name	Length	Req'd	Data Type	Comments
XML_STG_CTL_ID	10	Y	N	The unique identifier of the staging control record.
XML_STG_CTL_PARM	18	Y	A/N	Indicate the element in the WHERE clause whose value will limit the selection of records.
XML_STG_CTL_PVAL	250	Y	A/N	Use this to indicate the value used to limit selection of records.

Upload Staging Layout

You create an XAI upload staging record directly for each XML request you wish to make. The name of this table is **CI_XML_STGUP**. The following table describes each column on this table.

Column Name	Length	Req'd	Data Type	Comments
XML_STGUP_ID	15	Y	N	This is the unique identifier of the record. This value does NOT have to be a random number, but it does need to be unique.
APP_SVC_ID	20	Y	A/N	This is the foreign key to the application service table, and identifies the application service that is being requested.
COMPLETE_DTTM	26	N	Date/Time	Completion date and time.
CRE_DTTM	26	Y	Date/Time	Creation date and time.
RETRY_COUNT	3	Y	N	Number of retries to process the request.
XML_REQUEST	30,000	Y	A/N	The XML request document.
XML_RESPONSE	30,000	Y	A/N	The XML response document.
XML_STG_CTL_ID	10	Y	A/N	The id of the staging control record linked to this record.
XML_STG_STATUS_FLG	2	Y	A/N	Must be set to P (Pending) .

XMLUP-PR - Purge XAI Upload Objects

Completed XAI upload staging objects should be periodically purged from the system by executing the **XMLUP-PR** background process. This background process allows you to purge all **Completed** XAI upload staging objects older than a given number of days.

We want to stress that there is no system constraint as to the number of **Completed** XAI upload staging objects that may exist. You can retain these objects for as long as you desire. However we recommend that you periodically purge **Completed** XAI upload staging objects as they exist only to satisfy auditing and reporting needs.

XAI Upload Exception

A record is written to the XAI upload exception table for every XAI upload staging record that is in error.

To view the messages associated with the exception records, schedule the **TD-XAIUP** background process. This process generates a To Do entry for every record in the XAI upload exception table.

After correcting the cause of the error, drill into the [XAI Upload Staging](#) page and change the status from **Error** to **Pending** and the system will attempt to process the record again.

XAI Download Staging

XAI Download Staging - Main

To view individual XAI download staging records associated with a notification download staging record, open **Menu > Integration > XAI Download Staging**.

Description of Page

XAI Download Staging displays information about the record.

The **Download Staging ID** and **XAI Route Type** are the primary identifiers for this record.

The **XAI Download Status** is displayed. The values are **Pending**, **Complete**, **Canceled** or **Error**. When a record is in **Error**, it is displayed on the [XAI Download Exception](#) table.

Values from the NDS record associated with this XAI download staging record are displayed including **Service Provider**, **Notification Download Type**, **NT Download Status Flag**, **Retry Count** and the **Context** collection.

XAI Download Staging - Request

To display the XAI Request built by the download staging receiver, open **Menu > Integration > XAI Download Staging**.

Description of Page

The **XML Request** built by the download staging receiver is displayed.

XAI Download Staging - Response

To display the Response to this request, open **Menu > Integration > XAI Download Staging**.

Description of Page

The **XML Response** to this message is displayed.

XAI Download Exception

A record is written to the XAI download exception table for every XAI download staging record that is in error.

To view the messages associated with the exception records, schedule the **TD-XAIDN** background process. This process generates a To Do entry for every record in the XAI download exception table.

Maintaining Your XAI Environment

This section describes various tools provided to enable your XAI administrators to more easily maintain your XAI environment.

XAI Submission

This page exists for testing purposes. It allows you to create an XML request document and submit it to the system, to ensure that the XML has been built correctly.

XAI Submission - Main

To submit an XML document for testing, navigate to **Admin > XAI > XAI Submission** and navigate to the main tab.

Description of Page

This page is used to test XML schemas, which are defined for the XAI tool. Enter an appropriate XML document in the **XML Request** field. Typically, you define the XML schema using the schema editor in the XAI application. Then you would copy and paste the document here, then modify the schema to enter actual data for testing purposes.

When you have entered the document, choose Save to submit this document to the system. Note that this request information is not saved anywhere. It simply calls the system with the appropriate service name and executes the XML request.

Navigate to the Response tab to view the response.

XAI Submission - Response

To view the response to a XML document for testing, navigate to the response tab.

Description of Page

After choosing Save on the main tab to submit a test for an XML request, the response to your request is displayed in the **XML Response** text box.

Additional XAI Tools

This section introduces some additional tools to help you maintain your XAI environment.

XAI Service Export

The service export page allows you to export the definition of an XAI Inbound Service to a file. This function may be helpful if you need to copy the definition of this service to a separate environment. To export a service, open **Admin > XAI > XAI Service Export**.

Description of Page

Upon entry into this page, you are provided with the current list of **XAI In bound Services** and their **Description** s. Use the **XAI In Service Name** search field to find the XAI service that you would like to export.

Use the **Export?** column to indicate which XAI service(s) you would like to export. Once you have selected your services, choose **Save**.

NOTE: If multiple services are selected, they are exported together into the same output file.

You are presented with the standard File Download dialogue where you can open or save the file.

XAI Service Import

The service import page allows you to import the definition of an XAI Inbound Service from a file into the XAI service table. This function may be helpful if you need to copy in the definition of this service from a separate environment. To import a service, open **Admin > XAI > XAI Service Import**.

Description of Page

Upon initial entry into this page, you are provided with an input field, where you can enter the file name to import. Click **Browse** to search for the desired file in a directory.

Once the file is identified, click **Read File**, to read in the contents of the file.

NOTE: The format of the file must include tags indicating the column names for XAI Inbound Service table along with the values of the columns. For an example of how the format should be, simply go to the [XAI Service Export](#) page, export a Service and view the format of the resulting file.

Once the file has been read in, the list of XAI services found defined within the file is displayed in the Import grid, identified by their **XAI In Service Name** and **Description**. In the **Import?** column, indicate which services to import.

If a service with this service name already exists in the table, you must check the **Overwrite Existing** switch in order to indicate that the imported file information should replace the current service. An [XAI Inbound Service](#) that is provided as part of the system (i.e., with an owner of **Base Product**) may not be overwritten.

Click **Save** to proceed with the import. If any problems are found, information is displayed in the **Message Text** column.

XAI Command

Use the XAI Command page to send commands to the XAI and MPL server. To execute a command, open **Admin > XAI > XAI Command**.

Description of Page

The following operator commands may be sent to the XAI server. For each of these commands, you may check **Also Sent to MPL URL**, in which case, the command is also sent to the MPL server. You need to indicate the URL of the MPL server.

Display Registry Use this command to display the current registry information that the XAI instance is running with.

Refresh Code and Description This is related to an attribute in the schema editor where you may indicate that the description of a control table code should be returned along with the code itself. This information is kept in cache memory in the server. As a result, changes made to descriptions have no effect on the runtime server. This command clears the cache of control table codes and descriptions accessed by the server.

Refresh Registry The registry contents are kept in cache memory in the server. As a result, making changes to the registry control tables has no effect on the runtime server. Use this command to refresh the contents of the registry cache with the new values in the registry control tables. The command reloads all registry control table data into the server.

Refresh Schema Schema definitions are stored in cache memory on the XAI server. As a result, modifying a schema definition has no effect on the runtime server. To refresh schema definitions, use the Refresh Schemas command.

Refresh Service Service definitions are stored in cache memory on the XAI server. As a result, modifying an XAI inbound service definition has no effect on the runtime server. To refresh service definitions, use the Refresh Service command. You are prompted to indicate which service to refresh.

Refresh XSL XSL Transformation script definitions are stored in cache memory on the XAI server. As a result, modifying an XSL transformation definition has no effect on the runtime server. To refresh XSL transformation definitions, use the Refresh XSL command.

Trace On Use this command to start the XAI server trace.

Trace Off Use this command to stop the XAI server trace.

XAI Trace Clear Use this command to clear the contents of the trace file.

XAI Trace Swap Use this command to rename the current trace file by appending the date and time to the end. A new trace file is then created with the name as defined in the [Message option](#) page.

The following operator commands can be sent to the MPL server. You must set the URL of the MPL server first.

MPL Refresh Executer Executer definitions are stored in cache memory. As a result, adding or modifying executer definitions has no effect on the runtime server. Use this command to refresh executer definitions. You are prompted to indicate the executer to refresh.

MPL Refresh Receiver Receiver definitions are stored in cache memory. As a result, adding or modifying receiver definitions has no effect on the runtime server. Use this command to refresh receiver definitions. You are prompted to indicate the receiver to refresh.

MPL Refresh Sender Sender definitions are stored in cache memory. As a result, adding or modifying sender definitions has no effect on the runtime server. Use this command to refresh sender definitions. You are prompted to indicate the sender to refresh.

MPL Start Receiver Use this command to start a particular receiver. You are prompted to indicate the receiver to start.

MPL Stop Use this command to stop all MPL activity. It stops all receivers and waits for all executers and senders to complete.

MPL Stop Receiver Use this command to stop a particular receiver. You are prompted to indicate the receiver to stop.

MPL Trace On Use this command to start the MPL server trace.

MPL Trace Off Use this command to stop the MPL server trace.

When you have chosen the appropriate command and indicated any extra information, click **Send Command** to send the command to the server(s).

If you have sent a command to the XAI Server, then the bottom portion of the screen displays the response in the **XAI Response**. If you have sent a command to the MPL Server, then the bottom portion of the screen displays the response in the **MPL Response**. If you have sent a command to both servers, the bottom portion of the screen displays both responses.

MPL Exception

The MPL Exception table is used by the MPL to keep information about requests that resulted in a system error. These are errors that occurred inside the MPL. For example, if the MPL fails to send a request to XAI (maybe WebLogic is down), this is a system error, which would be logged in the MPL exception table.

There are errors that are defined recoverable. This means that the MPL will retry the action that failed, according to the parameters it received.

Server Trace

The XAI server traces every request and response. The requests/responses are written to a trace file on the server. The trace file may be viewed using the [Trace Viewer](#).

Starting the Trace

The log starts automatically based on definitions in the [Message Options](#) in the traceType and traceFile options. To manually start the trace:

- Navigate to **Admin > XAI > XAI Command**.
- Select the **Start Trace** command from the command dropdown
- Click **Send Command**

Stopping the Trace

- Navigate to **Admin > XAI > XAI Command**.
- Select the **Stop Trace** command from the command dropdown
- Click **Send Command**

Trace Viewer

Use the Trace Viewer utility to view the log file. The Trace Viewer is installed when you install the XAI client tools. It can be found in the XAI program group under Start/Programs.

Main Page

When the Trace Viewer starts, select a trace file to view. A trace file may be opened in one of two ways:

- To open a trace file directly from its location on the web application server, use the **File, Open HTTP** menu item and provide the appropriate URL.
- To open a trace file on the local/network file system use the **File, Open** menu item

Description of Page

Once a trace file is opened, it displays a list of all the requests on the left side including the **Service Name**, the **Start Time** and the **End Time**.

To display the XML contained in the request and response entries for a displayed request, select a request entry.


Filtering Options

Since the trace file may contain a very large number of messages, the trace viewer limits the number of messages that can be displayed. It does that by displaying messages traced within the last x number of **Minutes**, **Hours** or **Days**.

Use the **Max Messages** to limit the number of messages displayed.

NOTE: Default Note. By default, the Trace viewer displays the first **200** messages in the trace file.

To view only errors in the trace, check the **Show only Errors** option.

NOTE: Refresh Display. After changing any of the above filtering options, click the refresh button  in order to redisplay the request entries based on the new options.

The **First Message Found** field indicates the date and time of the earliest entry in the trace file.

Viewing as Text

To view the trace file as text rather than viewing each entry in its XML format, use the **View, As Text** menu option. The contents of the trace file are displayed in text format in a separate window.

Statistics Page

Use the **View, Statistics** menu item to view the statistic page, which displays performance statistics about the XAI services that were executed in the XAI trace file.

For each type of XAI Service and transaction type, it displays the following information based on the requests traced in the XAI trace file:

- The **Service Name** with the transaction type in parentheses
- The **Number of calls** for this service in the listed trace records
- The **Average** duration **Time** (in seconds)
- The **Max** imum duration **Time** (in seconds)
- The **Min** imum duration **Time** (in seconds)

NOTE: Requests Included in Statistics. Only requests falling in the time selection criteria and listed on the main log viewer are processed for calculating the statistics.

To display a Duration Chart for a particular service, check the Service. A chart such as the one below is displayed.

Integrations

This chapter provide high level information about product integrations supported for all products that use Oracle Utilities Application Framework.

LDAP Integration

Organizations commonly use a Lightweight Directory Access Protocol (LDAP) security repository as a source of security credentials. The system provides support for importing users and groups from an external LDAP repository into the product to populate Users and User Groups in the system. Once imported, all user and group functions are available. You can resynchronize your LDAP users and groups at any time.

NOTE: Import only. The system currently supports importing LDAP information. Exporting users and groups from the system to LDAP is not provided.

NOTE: Additional configuration. When importing new users and / or groups, additional configuration is needed in the base product. For example, after importing a new user group and its users, the user group configuration should be updated to define the valid application services for the user group. After importing a new user, additional configuration may be needed on the user such as valid To Do Roles, valid Home Page, etc.

FASTPATH: Refer to [Defining Security and User Options](#) for more information about the application security and what it controls.

This section the functionality provided in the framework application that supports LDAP. Refer to the *LDAP Integration* white paper for more information about typical steps related to the full integration.

LDAP Integration Overview

This topic provides a high level overview of the integration process.

At a high level, the base product provides a process to import user group and / or user definitions from and LDAP repository. This is a one way integration.

- When importing a user, if it is not already found in the system, it will be added; otherwise its attributes will be updated according to the imported information.
- When importing a user group, if it is not already found in the system, it will be added; otherwise its attributes will be updated according to the imported information.
- When importing a user, its user group links will be updated as per the information in the import file. In addition, if there are any user groups linked to the user that are not found system, they will be added (however, the other users linked to that group in the LDAP repository will not be added as part of this step).
- When importing a user group, its user links will be updated as per the information in the import file. In addition, if there are any users linked to the user group that are not found system, they will be added (however, the other user groups linked to that user in the LDAP repository will not be added as part of this step).
- The import will not cause any deletions of the User or User Group to occur.

A Batch Process Initiates the Import

A batch process is used to initiate the import of information from the LDAP repository. **F1-LDAP** may be submitted ad hoc or may be set up in a scheduler to periodically re-sync the information from the LDAP repository into the application.

The batch process uses parameters to define how to connect to the LDAP repository. In addition, parameters are used to indicate which user or group is being imported.

Adjusting Data to Import

The system provides several mechanisms for adjusting data that is being added to the system:

- There is an **LDAP Import Preprocess** algorithm plug-in spot on the [installation](#) record. Algorithms plugged in here are called by the batch process prior to the add or update of any records. It may be used to make adjustments to the data before doing updates in the application.
- Specifically for creating or updating Users, the **F1-IDMUser** business object is used to add and create the user. The standard BO Preprocessing algorithm plug-in spot may be used to adjust data prior to creation.
- The LDAP mapping file supports some attributes to perform simple modifications to data.
 - The **transform** attribute supports values to truncate values or to convert data to upper case.
 - The **autoGenerate** attribute is specific to the User ID field. Setting this to true will trigger code that will automatically populate the User ID based on the user's name. Refer to [LDAP Mapping](#) for more information.

Performing Additional Processing After Import

The system provides a plug-in spot on the [installation](#) record called **LDAP Import**. Algorithms plugged into this spot are called after users or user groups have been added or updated. It may be used to perform any extra processing that may need to be executed.

In addition, for any additional processing related to the creation or update of a User, the standard [Business Object plug-ins](#) may be used for the **F1-IDMUser** business object which the LDAP batch process uses to create or update users.

Configuring LDAP Integration

To interface the LDAP based security repository with the authorization component of the Oracle Utilities Application Framework product the following must be performed:

- The location and port number of the LDAP based security repository must be defined to in the JNDI Server.
- • The LDAP based security repository must be mapped to the Oracle Utilities Application Framework security model. This mapping is expressed as an XML file containing the LDAP query, rules and defaults used in the transformation.
- • The mapping file must be configured on the **F1-LDAP** batch job.

Define the JNDI Server

The first step in the configuration process is to define the location of the LDAP based security repository server so that the interface can connect to the physical attributes of the interface. This is done by creating a [JNDI Server](#).

NOTE: The LDAP server is strictly not a JNDI source but is treated as a JNDI source for the integration.

Enter a reasonable JNDI Server name and description.

Populate the **Provider URL** using the format **ldap://<hostname>:<portnumber>** where **<hostname>** is the host of the LDAP server and **<portnumber>** is the port used for the interface.

For the **Initial Context Factory**, the interface uses the standard **com.sun.jndi.ldap.LdapCtxFactory** provided with java for the LDAP interface. If your vendor supplies a custom context factory it may be used. Refer to the documentation provided with your LDAP based security repository for further information.

Define Mapping

The critical component of the interface is a file that describes the mapping between the LDAP based security repository and the system's security model. This file contains the mapping, rules and queries used by the LDAP batch program to provide the interface. The LDAP batch job includes the reference to the mapping file as a parameter. Refer to [LDAP Mapping](#) for more information on defining the mapping file.

Configure LDAP Batch Process

At this point, many parameters for the **F1-LDAPbatch control** can be updated with system wide configuration.

- JNDI Server, User and Password may all be configured appropriately. Note that it is recommended that the **Security** setting for the Password be set to **Encrypt**.
- The **LDAP Configuration File** should be populated with the name and location of the LDAP Mapping file.
- If the LDAP service has any limitation to the number of objects that may be imported, configure the **LDAP Query Page Size** parameter to enable querying.

NOTE: Group and User Parameters. The assumption is that the Group or User input parameters are specific to a given import request and as such would not be populated as part of a configuration step.

NOTE: L2 Cache. The LDAP Import batch process requires the L2 Cache to be disabled since it needs to perform some updates in the outside of the worker threads. Any environment using LDAP Import must set **spl.runtime.batch.L2CacheMode=OFF** in the **threadpoolworker.properties** file. It is recommended to run the LDAP import in its own dedicated threadpoolworker.

LDAP Mapping

An LDAP repository consists of multiple entries. Each entry represents an object in the directory that is identified by a Distinguished Name (DN) and may contain one or more attributes. In a typical LDAP repository there is usually an entry for users and an entry for groups. The connection between users and groups may be implemented in two different ways:

- The users belonging to a group are defined in a special multiple-value attribute on the Group entry.
- The groups to which a user belongs are defined in a special multiple-value attribute on the User entry.

The mapping between LDAP security objects and base security objects is stored in an XML document that can be processed by the LDAP import batch job. As part of setting up your system for LDAP import, you need to define this mapping. The base package provides a sample mapping file called **ldapdef.xml** that can be used as a starting point and changed per your business requirements and your particular LDAP repository.

Once you have defined the mapping XML document, this is configured as a parameter in the **F1-LDAP** batch job.

The XML structure:

- The **LDAPEntry** element maps the LDAP entries to system objects (User or Group). The mapping file must contain one and only one LDAPEntry element for User and one for Group.
- The **LDAPCDXAttrMapping** element within the LDAPEntry element maps attributes in the LDAP entry to attributes in the system object.
- The **LDAPEntryLinks** element describes objects linked to the LDAP entry. When mapping the user entity you need to describe how the groups the user belongs to are retrieved. When mapping the group entity you need to describe how the users contained in the group are retrieved.

The following table describes the attributes to define for each element.

Element	Attribute	Description
LDAPEntry	name	The name of the LDAP entry: - Group - User
	baseDN	The base distinguished name in LDAP for this entry.
	cdxEntity	The name of the base product entity to which the LDAP entry is mapped: - Group - User
	searchFilter	An LDAP search filter that is used to locate LDAP entries. A %searchParm% string in that filter is replaced by the value from the user or group parameter from the F1-LDAP batch job submission.
	Scope	Sets the scope of the search. Valid values are: - onelevel (the value normally used) - subtree
LDAPCDXAttrMapping	ldapAttr	The name of the LDAP attribute to be mapped. Note that this may be referenced more than once to allow one LDAP element to map to multiple base product elements. For example, if an email address should be used both for the Login ID and the Email Address.
	cdxName	The name of the base product attribute to be mapped. For User, this is the element within the F1-IDMUser business object. For Group, this is either the 'group' or the 'description'.
	default	The default value that will be assigned to the element referenced in the cdxName attribute when the following occurs: - The LDAP attribute contains a null or empty value - The LDAP attribute does not exist or is not specified. Default values are applied only when creating a new entity and are not applied to updated entities.
	autoGenerate	Set this to true in order to turn on auto generation of the user ID. If this is true, the system will define user id as <first initial of first name>+<last name> all uppercase, to a maximum of 8 digits. If an existing user is found for the generated ID, a number will replace the eighth digit (or be appended to the end). The system will increment the number until a unique ID is found.
	transform	Use this attribute to indicate if a transformation of the data should occur. Valid values: uppercase , truncate . Note that this attribute should not be used in conjunction with the autoGenerate attribute.
LDAPEntryLink	linkedToLDAPEntry	The name of the linked entity (User or Group). Use User when describing the Group entity. Use Group when describing the User entity.
	linkingLDAPAttr	The multiple-value attribute name on the LDAP entity that contains the linked entity.
	linkingSearchFilter	The search filter to be applied to retrieve the list of linked objects, for example: (&objectClass=group)(memberOf=%attr%) The search filter may contain the string % attr % that acts as a substitution string and is replaced at run time by the value of the attribute named "attr" of the imported entity. If the LDAP entry you are describing is a Group and the string is %name%, it is replaced by the value of the "name" attribute of the group you are importing. If the LDAP entry you are describing is a User and the string is %dn% it is replaced by the "dn" attribute of the User you are importing.
	linkingSearchScope	Sets the scope of the search. Valid values are: - onelevel (the value normally used)

Element	Attribute	Description - subtree
---------	-----------	--------------------------

Sample Mapping

The following XML describes a sample mapping. The example makes the following assumptions:

- The base product attribute **displayProfileCode** is defaulted to "NORTHAM" when adding a new user.
- The LDAP Group entry contains the list of users belonging to the group in the **departmentNumber** attribute.
- The groups to which a user belongs are retrieved by applying a search filter.

```
<LDAPEntries>
  <LDAPEntry name=" User" baseDN="ou=people,dc=example,dc=com" cdxEntity=" user" searchFilter=" (&
(objectClass=inetOrgPerson)(uid=%searchParm%)) ">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="uid" cdxName=" user" />
      <LDAPCDXAttrMapping ldapAttr="cn" cdxName="externalUserId" />
      <LDAPCDXAttrMapping cdxName="language" default=" ENG" />
      <LDAPCDXAttrMapping ldapAttr="givenName" cdxName="firstName" />
      <LDAPCDXAttrMapping ldapAttr="sn" cdxName=" lastName" />
      <LDAPCDXAttrMapping cdxName="displayProfileCode" default="NORTHAM" />
      <LDAPCDXAttrMapping cdxName="toDoEntriesAge1" default="30" />
      <LDAPCDXAttrMapping cdxName="toDoEntriesAge2" default="90" />
      <LDAPCDXAttrMapping cdxName="userEnable" default="ENBL" />
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntity="Group" linkingLDAPAttr="departmentNumber" />
    </LDAPEntryLinks>
  </LDAPEntry>
  <LDAPEntry name="Group" baseDN="ou=people,dc=example,dc=com" cdxEntity=" Group" searchFilter=" (&
(objectClass=organizationalUnit)(ou=%searchParm%)) ">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="name" cdxName="Group" />
      <LDAPCDXAttrMapping ldapAttr="description" cdxName=" Description" default="Unknown" />
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntity="User" linkingSearchFilter=" (&
(objectClass=inetOrgPerson)(departmentNumber=%distinguishedName
%))" linkingSearchScope="onelevel" />
    </LDAPEntryLinks>
  </LDAPEntry>
</LDAPEntries>
```

Oracle Identity Manager Integration

The *Oracle Identity Manager* product allows a site to centralize their user definitions and password rules to manage and deploy across the enterprise set of products. When an employee joins an organization, changes their name or departs an organization their security presence across an enterprise must be appropriately managed. Oracle Identity Manager allows for users to be provision and managed in a central location.

An integration is provided to allow the ability to create, maintain and remove users in the identity management product and sync those changes to the users defined in the application. The following sections provide additional details about the integration with respect to configuration steps required in an Oracle Utilities Application Framework based product. For more information about the configuration required in the identity management product, refer to the *Identity Management Suite Integration* white paper.

Template User Functionality

The user object in this product captures configuration used to control access but also preferences. The identify management product allows for extending the configuration to capture user configuration that is specific to this product. However, it does not support providing searches or dropdowns to select valid values. For example, to define the user's Home Page requires

the reference to a navigation option. To set up your business process such that the home page is configured when defining the user in the identity management product dictates that the security user types in the correct navigation option reference.

On the other hand, to define a minimal amount of user information in the identity management product may result in a two step process for defining users: first define them in the identity management product with the basic authentication details and setting system defaults for some important fields, then after submitting the new user to be added to this product, navigate to the [user](#) page in this product and fill in all the configuration that is specific to this product.

The product provides support for defining a template user that can facilitate the definition of users and reduce some of the challenges listed above. The concept is as follows:

- Define a template user for each broad category of users in the system. For example, Oracle Utilities Mobile Workforce Management may define the following template users: Dispatcher, Mobile Worker, System Administrator and Contractor. Each user would define the typical configuration for users of that type including the home page, the user groups, the To Do roles, the portal preferences, etc.
- When extending the configuration in the identity manager product, simply map the information that is unique to a user and in addition, define a field for the template user. For example, you may choose to only capture the Name (first and last), Email address and User IDs for the user along with its Template User (which is mapped to a user characteristic). Additional fields may be included for capture in the identity management product when defining new users as per an implementation's business needs. For example, if the organization covers multiple time zones, perhaps it is easier to define the user's time zone when defining the user in the identity management product.
- When the new user is uploaded to the system, the interface uses the user BO **F1-IDMUser** to create the user. The BO includes a preprocessing algorithm that looks for the existence of a template user (sent as a characteristic of type **F1-TMUSR**). All the information from the template user will be copied onto the new user record except for the information passed in from the identity manager. The template user is captured on the newly created user via a characteristic for information / audit purposes.
- Once the new user is created, its configuration can now be adjusted, if applicable. Note that the template user is only a tool used when adding a user. Updates to the template user will not "ripple" to all the other users that were created based on this template.

Configuring Template Users

Before configuring template users, all the administrative control tables that are part of User configuration must be defined, including time zones, display profiles, To Do roles, data access roles and user groups.

The next step is to define the user configuration for your system users. During this exercise, you will find that you have broad categories of users. But you will also see that within a given category of user there may be variations in the user privileges and preferences. For example, perhaps there are supervisors within the Mobile Worker role that have more security privileges than a typical Mobile Worker. In addition, there may be variations based on the attributes of the users themselves. For example, maybe your organization exists in multiple time zones and some of your workers are in one time zone and some are in the other.

At this point your security users that are designing their user provisioning procedures must decide the following:

- What information about a new user will be captured in the identity system (besides the expected information like Name, Email and the User IDs)? For example, for the case of multiple time zones, maybe the best solution is to capture the time zone when defining the user.

What information is defined on the template user and how many template users should be created to reduce the need for manual steps or additional data captured in the identity management system? In the case of multiple time zones, proliferating the template users to have one set for one time zone and another set for the other time zone may not make sense since this is one field that is different. However it may be reasonable to create additional templates in the case of variations in the levels of privileges for workers of a different category. So rather than template users for Dispatcher, Mobile Worker, System Administrator and Contractor, your organization may have template users for Dispatcher, Mobile Worker, Mobile Worker (Supervisor), System Administrator, Contractor (Short Term) and Contractor (Long Term).

What information is must be configured one the User record in the application after the user is added? If only a small number of users have a variation from other users, it may be that the easiest way to deal with those variations is to simply update those user records manually. Using the above examples,

- If your organization covers 2 time zones but only a small group of people work in one of the time zones whereas the bulk of the users are in the other time zone, the simplest procedure may be to define the template users for the main time zone and use that for the creation of all users. Then for the small group of users in the separate time zone, navigate to the User page to adjust the time zone after the record is added.
- If only a small number of Mobile Workers are supervisors with separate privileges, rather than defining a special template user for those type of workers, the simpler procedure may be to use the Mobile Worker template and then navigate to the User page to add the additional privileges to the supervisor users after the record is added.

To create a template user, navigate using **Admin > Security > User** .

- Define a User ID that will become the template user reference in the identity management system.
- Be sure to choose a **User Type** of **Template User**.
- Define all the information that should be copied onto a new user that references this user as a template user. Note that **Bookmarks** are not included in the data that is copied from a template user.

NOTE: There is configuration needed in Oracle Identity Management to capture the template user and any other information that the implementation has chosen to define in the identity management product when provisioning a new user. Refer to the *Identity Management Suite Integration* white paper for more information.

Batch Scheduler Integration

The Oracle Database includes an enterprise wide scheduler to simplify the scheduling of background processes. The scheduler is implemented by the *DBMS_SCHEDULER* package. The product provides an integration with the Oracle Scheduler to facilitate scheduling background processes shipped with the product.

NOTE: For details on the integration, refer to the *Server Administration Guide* which contains the API. In addition, refer to the white paper *Oracle Scheduler Integration* that provides guidelines for using this integration.

Data Synchronization

Your implementation may need to communicate certain data to external systems. This may be part of a data warehousing requirement or an integration effort. The synchronization process has two main parts. First, the change to the data must be detected and captured. Once that is accomplished, the next step is to manage the communication of that change to the external systems involved. The changes must be captured in chronological order so as to avoid systems going out of sync.

About Data Synchronization

The following sections describe general supported functionality in more detail using the logic supplied in the base business object **F1-SyncRequest**. Note that each edge application delivers an appropriate child business object for this BO for each specific sync scenario supported in that product. Some of the functionality below is accomplished using configuration on the parent BO delivered by the framework while other functionality may be delivered by the child BO. In addition, there may be more complex use cases supported by your specific product integration. Refer to your specific application's library of Sync Request business object along with the documentation related to your specific product integration for more information.

Capturing the Change

The base product uses the **Audit** plug-in spot on the maintenance object to allow for logic to be performed by the system when a change is detected to a record for that MO. The framework calls the algorithm defined on this plug-in spot in the event a change to the MO has been detected. Refer to the description of the plug-in spot on [Maintenance Object — Algorithms](#) for more information about when this plug-in is called.

The base product provides an change data capture algorithm **F1-GCHG-CDCP** that may be used by maintenance objects. This algorithm creates a Sync Request record for each changed record, capturing the MO code and the primary key, if it doesn't find an existing sync request for the same record (and the same business object) in the initial state. The sync request business object used is the one defined in the **Sync Request BO** option on the MO for the record that was changed.

Your specific product may also introduce additional Audit algorithms to cater for more sophisticated examples. Click [here](#) to see the algorithm types available for this system event.

When creating the sync request record, typically the Sync Request BO will have a pre-processing plug-in that captures a snapshot of the record's data prior to its change. This will be used in subsequent steps to verify that the external system needs to be notified of the change.

Confirming that a Sync is Needed

Once a sync request is captured, there are several steps performed prior to any information being sent to the external system.

NOTE: This section only highlights key steps. Please refer to the business object configuration, its lifecycle and algorithms for a thorough picture of the full functionality..

- When a Sync Request record is created, its initial state (**Pending**) is configured to be processed by a batch monitor. That way, records are added to the sync request table added throughout the day but all are processed together. The MO audit algorithm ensures that a new synch request is not created if a Pending record already exists for a given MO / PK combination (for the same business object). However, it is possible that a record for that MO / PK exists in a subsequent “non-final” (such as **Awaiting Acknowledgement**). This state includes a monitor algorithm to check for that condition and to skip transitioning if another record exists. This is done to ensure that the existing record is fully processed before this new record is processed.
- The next state of the lifecycle is **Determine if Sync Needed**. This step uses an algorithm to take a snapshot (called the ‘final snapshot’) of the data and compare it against the initial snapshot taken when the record was created. Based on the logic of the algorithm, it may decide to proceed (transition to **Send Request** or to discontinue (transition to **Discarded**).

Communicating to the External System

Once it is confirmed that the sync should occur, a message must be sent to the external system. The following points highlight the basic functionality.

- An algorithm linked to the **Send Request** state. The expectation is that this algorithm creates an outbound message that routes the information to the external system appropriately. The algorithm must determine the external system and outbound message type to use. Business objects for Sync Request support BO options to define the external system and outbound message type to use for this algorithm.
- Once the outbound message is triggered, the record transitions to the **Awaiting Acknowledgement** status. This state is used to hold the sync request from further state transitions until an acknowledgement is received from the external system. Note that this step relies on implementation of a response mechanism from the external system. It is recommended to implement a response as this helps control the chronological flow of information. The product supplies the business service **F1-UpdateSyncRequest** that transitions the sync request to either the next default state (in this case the **Synchronized** state) if a positive acknowledgement is received; or the state associated with the Rejection transition condition (in this case the **Error** state) if a negative acknowledgement is received. In addition, this state may be configured with a monitor algorithm that detects that a timeout limit has been reached.

- For records that enter the **Error** state, it is recommended to configure an algorithm that creates a To Do entry to alert someone of the problem. Refer to the integration documentation for more information. The state is already preconfigured with an algorithm to complete To Dos when exiting the state.
- The final state **Synchronized** is used to mark the successful synchronizations. However, for more complicated use cases, this state may be used to trigger some additional action. Refer to the documentation for your specific product integration for more information.

Maintaining Sync Requests

The system provides a Sync Request portal that is used to view the in progress or completed sync request records.

The menu location of the portal depends on your specific edge product. It may be in a Data Synchronization menu or perhaps in the Batch menu. You are brought to a query portal with options for searching. The options may differ based on your specific product.

Once a sync request has been selected, you are brought to the maintenance portal to view and maintain the selected record.

An **Actions** zone may appear to display specific actions. Alternatively, the actions may be displayed directly in the display area of the **Sync Request** zone.

The **Sync Request** zone provides basic information about the sync request record.

Depending on your specific product additional zones may appear.

Analytics Integration

The framework provides several building blocks and tools that the edge applications may use to implement integration with the Oracle Utilities Analytics product (referred to in this section as the analytics product). The following sections provide more information about this functionality.

About Analytics Integration

The following sections describe the type of configuration supported in your product to integrate with the analytics product. Refer to the Oracle Utilities Analytics documentation for more information.

Master Configuration

Edge applications that include an integration to the analytics product typically include a master configuration record that captures information needed for the extract, load and transformation step, such as extract parameters. These records are provided by the specific edge products and may be viewed and maintained on the [master configuration](#) portal.

Note that your specific edge application may deliver an **Analytics Configuration** portal that displays the information from the master configuration record along with other analytics related configuration.

Bucket Configuration

The analytics product provides support for defining a set of ranges, each representing a bucket for which extracted measures can be grouped and classified under the relevant bucket. The framework product provides support for viewing and defining the buckets. Refer to [Bucket Configuration](#) for more information.

Characteristic Mapping

The product provides objects to allow mapping configuration of characteristics in the product to user defined fields on dimensions in the analytics product. Refer to ETL Mapping Control for more information.

Change Data Capture Using Sync Request

Depending on the specific edge application and version you are using, there may be components of the integration that use Sync Request for the change data capture step. If that functionality applies to your implementation, the following points highlight how to get more information:

- Refer to the administration guide for Oracle Utilities Analytics to confirm if your product integration is using Sync Request for any change data capture functionality.
- Review the Sync Request Business Objects provided by your product for analytics integration.
- Refer to [Data Synchronization](#) for a high level understanding of the process.

Maintaining Bucket Configurations

Several key performance indicators in the analytics product look at measurement values (for example: the age of an asset or the age of debt) classified into a number of pre-defined groups also known as buckets. The overall metric can then be reported by the different buckets and allow various analyses.

For example, the age of an asset can be classified into the following buckets:

- Less than 6 Months
- 6-12 Months
- One Year and Older

The age of debt, also known as arrears can be classified onto the following buckets:

- 0-30 Days
- 30-60 Days
- 60-90 Days
- 90+ Days

The definition of the buckets is extracted to the Business Intelligence data warehouse, to be used as dimensions.

Bucket Definition Considerations

Each type of bucket is defined using a bucket configuration Business Object. The bucket definition considerations and/or rules will vary based on the bucket configuration business object used. The business objects available are driven by your specific product. For a list of available bucket configurations business objects, navigate to the business object page and view the business objects for the Bucket Configuration maintenance object.

Setting Up Bucket Configurations

To maintain the bucket ranges for the bucket configuration(s) applicable to your product, open **Admin > Analytics Configuration > Bucket Configuration**.

You are taken to the query portal where you can search for an existing bucket configuration. Once a record is selected, you are brought to the maintenance portal to view and maintain the selected record.

NOTE: Your specific product may also include an Analytics Configuration portal that displays the list of existing and potential bucket configuration records, allowing you to drill into this page to view the record in detail.

The **Bucket Configuration** zone provides basic information about the bucket configuration.

For more information about the elements supported refer to the zone's help or to the relevant analytics integration documentation for your product.

ETL Mapping Control

If your implementation would like to mapping characteristic data in your application to user defined fields on dimensions in the analytics product, the ETL mapping control provides configuration to support this.

The ETL mapping control relies on configuration in the Allowed Target Dimensions [extendable lookup](#). The extendable lookup is used to define each Target Table in the analytics product that has one or more user defined fields that may be populated with characteristic values. It also defines the valid characteristic entities that may act as the source for the characteristic data.

NOTE: The framework does not provide any values in this lookup, but edge products that support mapping provide values in this lookup. Please refer to your specific product's integration chapter and refer to the Oracle Utilities Analytics documentation for more information.

Setting up ETL Mapping Control

Use the ETL mapping control to define how to populate each target table (dimension) and target column (user defined field) by indicating the characteristic entity, characteristic type and for sequence-based characteristic, the sequence number.

NOTE: For sequence based characteristics, if the source entity captures multiple characteristic values for a given characteristic type, each characteristic type and sequence combination must be mapped to a specific target table and target column.

NOTE: Only **Adhoc** and **Pre-defined** characteristic types are supported.

To maintain the ETL mapping control records, open **Admin > Analytics Configuration > ETL Mapping Control**.

This is a standard [All-in-One portal](#).

Refer to the inline help text for more information.

Business Flags

It is possible that information detected in one product may be useful or even critical to share with another product. The framework provides functionality for receiving information from an external system that acts as a type of flag or alert that may need investigation. This allows any system to store detected business flags in a common way and share that information with one or more other systems.

About Business Flags

The following is an example of a use case for business flags. Imagine that DataRaker highlights potential theft of service at a certain location. That product may initiate a business flag alert to various products owned by the implementation with a recognized "standard name" for the business flag, such as "TAMPER".

- If Oracle Utilities Meter Data Management receives this business flag, it may initiate a service investigation monitor.
- If Oracle Utilities Meter Workflow Management receives this business flag, it may initiate a service investigation activity.
- If Oracle Utilities Customer Care and Billing receives this business flag, it may initiate a hold on billing for that location.

Note that the framework product supplies basic functionality to support logic that is common to all edge applications that implement business flag functionality. However it is the individual edge applications that supply more specific functionality (business objects and algorithms) for specific use cases, if applicable.

The following sections highlight functionality supported for business flags in the framework. Refer to the edge application product documentation for more details for supported use cases.

Standard Name

To ensure that Business Flags are universally understood across all edge applications and to simplify integration each Business Flag will have a Standard Name. This is a name that is used by all the products when sending information to each other. That way, if DataRaker product sends Oracle Utilities Meter Data Management a “TAMPER” business flag, it should result in the same functionality as when Utilities Customer Care and Billing sends a “TAMPER” business flag.

Business Flag Standard Names are defined using an extendable lookup. In addition to standard extendable lookup fields, the standard name also references a category. In addition, the lookup supports defining one or more external names, for cases where information is communicated from an external system that does not send the expected Standard Name.

The framework does not deliver any standard names or category values. Refer to your specific edge application products to verify if any standard names or categories are delivered. Your implementation may configure appropriate standard names and categories based on your business rules.

Business Flag Type Defines Behavior for a Standard Name

Although the definition of the business flag standard names should be universally understood by the various integrated products that support them, each individual product defines what should occur when a business flag with a given standard name is created. This is configured using a business flag type. Only one active business flag type may exist for a given standard name. Business flags that are received from an external system will define the standard name, but will not have knowledge of the specific business flag types defined. The business flag type is determined based on the standard name.

The business flag type defines the business object to use when creating the resulting business flag record. The business object defines the lifecycle of a resulting business flag record.

Business Flag Type Algorithms

The business flag type includes support for algorithms. This allows for an implementation to define a common business object that may be used for different business flag types (if a common lifecycle is followed) but allow for different functionality to kick in depending on the business flag type.

The product supplies a plug-in spot for **Additional Processing** that may be invoked by a business flag that enters the Additional Processing state. Refer to your product’s library of business objects to determine if there is an Additional Processing state that supports calling algorithms on the business flag type.

Click [here](#) to see the algorithm types available for this system event.

Objects Linked to a Business Flag

There are two types of links between an object in the system and a given business flag.

Affected Entity

Each business flag is associated with a single record in the system that is considered the “affected entity” or the entity that the business flag is associated with. The affected entity is defined by the specific business objects designed for the use cases supported by your edge product. For example, many utility base products may configure service point as the affected entity for its business flag use cases. Each business flag created is linked to a specific service point. Linking a specific entity to each business flag allows for algorithms to trigger functionality for that entity such as an investigation order. In addition, algorithms may be implemented in other business process areas that look for the existence of a business flag and act accordingly.

Related Objects

The business flag supports linking one or more related objects to a business flag to make it easier to trigger functionality or for impacted business processes to look for business flags. For example, when creating a business flag for a given service point, it may be useful to link all the accounts that are currently linked to the service point. Then, if an account oriented process should check for a business flag, it can look directly for a business flag linked to the account in its related object.

Impacted Business Process

The product supplies support for associating one or more impacted business processes to a business flag type. This configuration is used when functionality for that business process is impacted in some way based on the existence of a business flag of a given type. For example, maybe some process is put on hold when a certain type of business flag exists.

Note that configuring a business process on business flag type is not enough to trigger any impact on that business process when a business flag exists. There must also be some logic implemented in the business process functionality itself that knows to look for a business flag for a given record that is configured to impact the business process.

The definition of the business process is at the discretion of the edge application that supplies functionality to support this. For example, the business process could be defined as something broad such as “billing” or could be something more granular such as “billing estimation”. The system supplies an extendable lookup to use for configuring the supported business processes. Refer to the values of the business process extendable lookup in your edge application or to the edge application specific Business Flag documentation for more information about supported business processes.

Dates

A business flag supports two dates: a Business Flag Date/Time and a Business Flag **End** Date/Time

- The business flag date / time is required for all business flags. For some types of business flags only one date is needed.
- For business flags that have a start and end period, the business flag date/time acts as the start date and the other field is the end date.

For a business flag that has a date range, it may be important for functionality implemented for [impacted business processes](#). How the process treats the date will depend on its functionality.

- For some processes, the business flag is essentially expired after the end date has passed. This applies to impacted processes that are only looking at the current status of data in the system. For example, collection processing could be held if there is a business flag currently in effect (where the current date is within the date range). It would never look at historical business flags.
- For some processes where historical data may be relevant, a business flag effective during that same historical period may impact the process. For example if a business flag denotes an outage event for a given time period, perhaps estimated consumption should never be calculated for that time period.

Note that because business flags have a status, the design for the lifecycle of the business objects for the above effective dated use cases must carefully consider the states. For business flags that are considered expired after the end date passes, the BO lifecycle may be designed to transition to a final state after that date such that the record is no longer included in active processing. For business flags that continue to impact processing for a historic period, the BO lifecycle may be designed to remain in a non-final state such that it is clear that the record is still applicable.

Creating Business Flags

Business flags may be created in a system for one of the following reasons:

- A message is received from an external system that initiates the creation of a business flag. In this case, logic in the external system has detected some situation that this product is being alerted about.

- Business logic in this product detects a situation that should be investigated or should act as a flag. In this scenario, there may not be any integration needed depending on the business rules.
- Business logic in this product detects a situation that another integrated product should be alerted about. In this scenario, the business flag record is used to send out information to the integrated product.
- A user manually creates a business flag based on knowledge of the affected entity. For example, a customer service representative may create a business flag as a result of contact with the customer.

Creating a Business Flag from a Web Service

The system supplies an inbound web service Business Flag Sync Driver (**F1-BusinessFlagSync**) that may be used for an external system to initiate (or update) a business flag. It references a service script whose ultimate responsibility is to determine the appropriate Business Flag Type, based on the standard name or external standard name, and therefore the appropriate business object for the new business flag. Because different products may have different logic related to creating a business flag, the service script calls another service script linked to the maintenance object using the Business Flag Sync MO option.

The framework does not supply a Business Flag Sync service script, however individual edge applications supply a service script based on the use cases it supports out of the box.

NOTE: For products that are continuing to use XAI for external messages, the product also includes an XAI inbound service for the same Business Flag Sync Driver service script. Note that the product recommendation is to discontinue use of XAI and use [inbound web services](#) instead.

Error Handling

If there is a problem in trying to create a business flag based on incoming information, the Business Flag Sync Driver service script creates a special business flag record using the Business Flag Error Business Object. This is also configured on the maintenance object as an option. The framework product supplies the business object Business Flag Error (**F1-BusinessFlagError**) for this functionality. Refer to the business object description and configuration for more information.

Confidence

There may be use cases where a condition is suspected, but not confirmed. The originating system should be able to assign a “confidence” level to the business flag.

For example, DataRaker through aggregating and analyzing large amounts of data identifies potential revenue protection issues that need investigation. It triggers business flags with a **Suspected** confidence.

An application receiving this business flag may adjust the confidence to either **Confirmed** or **Rejected**. That update can be communicated to the other products that received the same business flag.

Note that the application that receives a business flag is responsible for acting on the value based on business rules.

Because a utility implementation may have multiple applications installed that support business flags, the following guidelines are suggested for designing where the confidence flag should be updated.

- If Oracle Utilities Service Order Management is implemented, it has the responsibility of updating the confidence flag and communicating the update to other products.
- Otherwise, the assumption is that Oracle Utilities Customer Care and Billing owns field work orchestration and that it will have the responsibility for updating the confidence flag and communicating the update to other products.

No product logic is provided to enforce the above suggestions, however, the business objects supplied by the different edge applications will support the recommended implementation.

Business Flag Updates from External System

When the product that is responsible for updating the Confidence flag makes a change, it should initiate an outbound message to alert other products. On the receiving side, the same inbound web service and Business Flag Sync service script is responsible for the update. Refer to [Creating Business Flags](#) for more information.

Setting Up Business Flag Configuration

The following topics highlight the general configuration steps required to use business flag functionality. Your specific product may supply additional functionality to support specific use cases for business flags. Refer to your specific product's documentation and the library of business objects supplied for Business Flag in your product for more information.

Standard Name Category Characteristic Type

Define one or more categories for grouping your standard names into logical business groupings.

Navigate to the [Characteristic Type](#) page. Search for and select the Business Flag Category characteristic type (**F1–BUSFC**).

Define desired category values and descriptions to be used for the standard names.

Business Flag Standard Name Lookup

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Business Flag Standard Name** business object.

Define values that are recognized in the external systems that your implementation is receiving business flag details from.

Define a **Category** for the standard name that is appropriate for your product. Note that the category does not have to be in sync with standard name definitions in external products.

Refer to the inline help for more information about configuring this object.

Business Process Lookup

If your specific product supports configuring business processes that may be impacted by the existence of a business flag, they are defined as an extendable lookup.

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Business Process** business object.

Integration Configuration

The following points highlight configuration required to support receiving business flag information from an external source:

- Define a record for each [External System](#) that the product may be receiving business flag records from. This should be a value known by the external system and provided when new business flags are sent to this product.

When this product should initiate business flag information to be sent to an external system, configure one or more [Outbound Message Type](#) records. For each one, update the External System to configure how each outbound message type is communicated to the external system.

Defining Business Flag Types

Refer to [About Business Flags](#) for an overview of business flag functionality.

To maintain the business flag types applicable to your product, open **Admin > Integration > Business Flag Type**.

This is a standard [All-in-One portal](#).

The information captured on the business flag type depends on the business objects supported by your product. Refer to the inline help text for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [F1_BUS_FLG_TYPE](#).

Maintaining Business Flags

This section describes the functionality supported for viewing and maintaining business flags.

Refer to [About Business Flags](#) for an overview of business flag functionality.

Navigate using **Main > Integration > Business Flags**. You are brought to a query portal with options for searching for business flags.

Once a business flag record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Business Flag** zone provides basic information about a business flag. Refer to the inline help for more information.

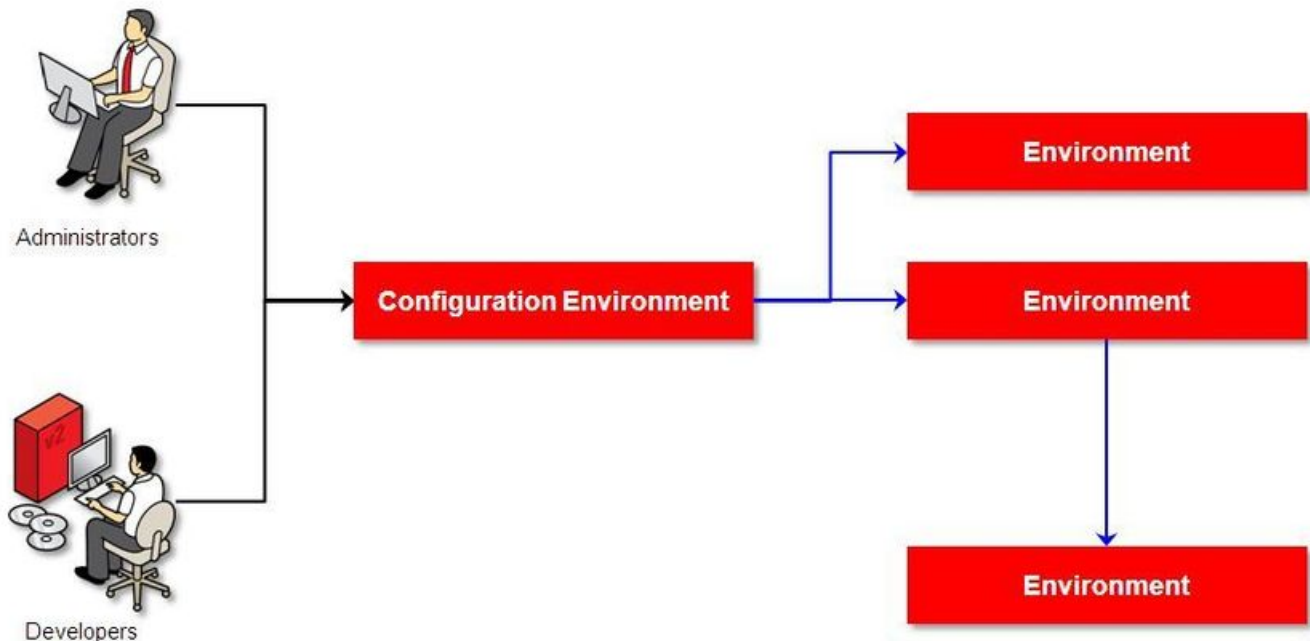
Configuration Migration Assistant (CMA)

This chapter describes the Configuration Migration Assistant (CMA), a facility to enable the export of customer-owned configuration data from one environment to another.

CAUTION: This chapter is intended for users responsible for testing configuration data and performing "what if" analysis in non-production databases.

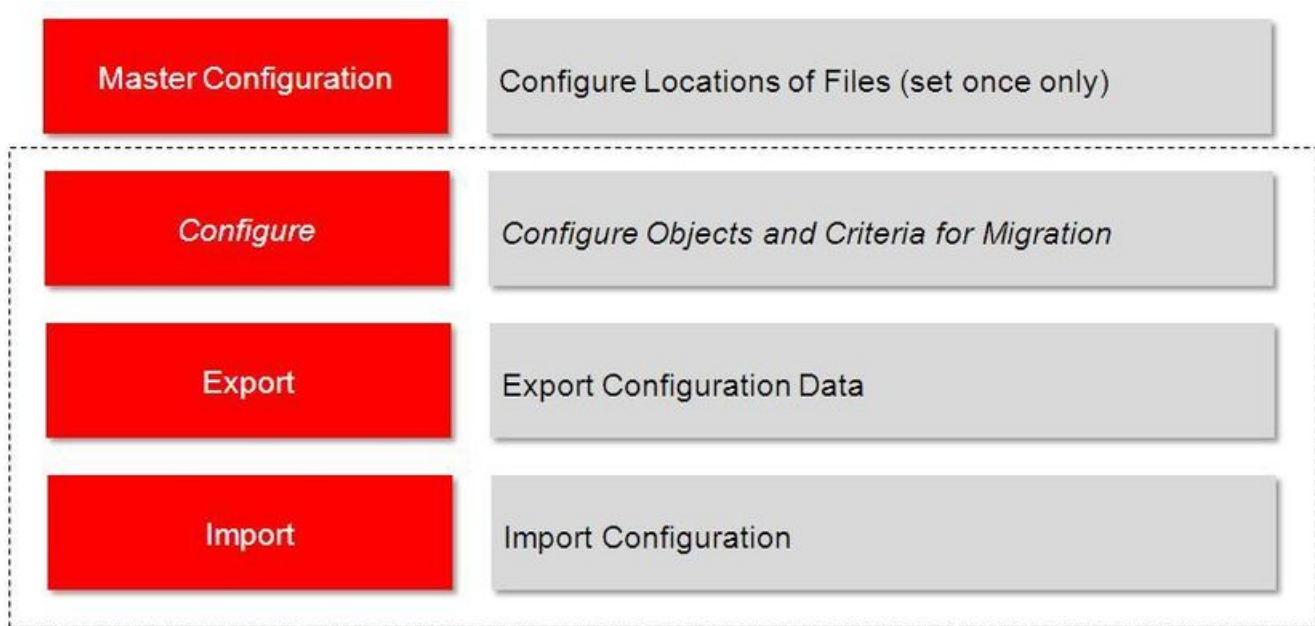
Understanding CMA

The Configuration Migration Assistant (CMA) provides implementers with a flexible, extensible facility for migrating configuration data from one environment to another (e.g., from a development environment to a production environment). Data is exported from the source system to a file. The file can then be checked in to a version control system for reuse, or can be immediately imported into the target system and applied.



NOTE: As used in this chapter, *source* systems are those on which export-related activities are conducted and *target* systems are those on which migration updates are to occur. The two system preparation tasks described in [Migration Assistant Configuration](#) must be performed on both source and target systems.

The CMA Process Flow



The high-level CMA process comprises the following tasks and flows. Each is described in more detail later in this section.

- **Configuration** steps are used to define the data to migrate. This task is performed on the source system and may be defined at any time. Note that the products provide base delivered configuration that may be used as is or used as a template for building more specific configuration for a given implementation.
 - Specify the source and target paths and a file extension for import and export data files in the **Migration Assistant Configuration** master configuration record.
 - Each type of record that may be copied requires a **Migration Plan**. The migration plan is used to identify the maintenance object (MO) for the record (using a business object) and allows for instructions to specify related records that may be included in the migration. Multiple migration plans may exist for a given maintenance object if there are different requirements for migrating records in that MO under different circumstances.
 - The primary instruction in the migration plan could define the physical BO of the record. This signals to CMA that all records for the MO are eligible to be migrated.
 - The primary instruction may define a specific BO, in which case only records that reference that BO in its parental hierarchy are considered.

NOTE: Refer to [Understanding the BO Filtering Process](#) for more information.

- Subsequent instructions may include references to related records. There may be some circumstances where a migration should include subsequent records and some circumstances where they shouldn't based on requirements.
- A **Migration Request** is used to define a set of migration plans to be included together in a single migration attempt. In addition, the migration request may define selection criteria for each migration plan to limit the migration to a subset of data for each MO. Selection may be defined using SQL, an algorithm or explicit primary key values.

FASTPATH: For more information, refer to [CMA Configuration](#).

- The **Export** process includes all the steps needed to select records to be exported from the source environment and create the export file.
 - A **Migration Data Set Export** object is created for a specific migration request.
 - The lifecycle of the Migration Data Set Export business object includes algorithms that select the appropriate records according to the migration request, determine dependencies between records to build groupings of related objects and create the export file.
 - Because there may be a large volume of data in the export, many of the steps in the lifecycle of the migration data set export are executed via the **Migration Data Set Export Monitor**.

FASTPATH: For more information, refer to [Exporting a Migration](#).

- The file created by the export, which is a BINARY file, needs to be transferred from the export directory to the import directory. The transfer needs to be done in such a way as to preserve the file structure. Refer to [Migration Assumptions, Restrictions and Recommendations](#) for more information.
- The **Import** processes include all the steps needed to read an imported file, compare the data in the file to the data in the target, review the proposed changes and apply the updates. The following is a high level overview of the process.
 - A **Migration Data Set Import** object is created for a given file to import.
 - The next step reads the import file and creates **Migration Transactions** and **Migration Objects** based on the information in the import file. A migration object is created for every maintenance object record to potentially be created or updated. The migration transaction is a grouping record that groups together related migration objects.
 - The next step is for the objects to **Compare** the data being imported against the data for that record in the target environment. If it is found that the migration object is new or represents a change to the existing record in the target environment, appropriate SQLs are generated. If no changes are found, the object is marked “unchanged” and doesn’t progress.
 - Once all the objects are compared, the user may review the objects for acceptance or rejection.
 - When the migration objects are all accepted or rejected, the next step is to apply the objects and update the target environment.

FASTPATH: For more information, refer to [Importing and Applying a Migration](#).

Migration Assumptions, Restrictions and Recommendations

The following sections describe miscellaneous topics that are important to learn with respect to CMA.

Type of Data Migrated

CMA is designed to migrate configuration data.

The comparison step of the import process will generate appropriate insert or update SQL statements for the following data found in the export:

- Configuration data in a maintenance object with no owner flag. This is purely implementation data.
- Configuration data in a maintenance object with owner flag, where the owner is **Customer Modification**. For example, implementer-specific business objects.

- Configuration data in a maintenance object with owner flag, where the main record is owned by the product but where a child record exists with an owner of **Customer Modification**. For example, implementer-specific algorithms added to a base owned business object.
- Customizable fields in a record that is owned by the product. For example, the priority of a based owned To Do type.

Data with System Generated Primary Keys

The tool provides support for administrative data with system-generated primary keys. The logic relies on the maintenance object to use a method that looks at other attributes of the record (considered a “logical key”) to detect whether the record being migrated already exists in the target region or not. The examples in this section will use the Attachment maintenance object as an example. Common attachments are considered administrative data. The attachment MO uses the file name and the creation date as the “logical key”.

Imagine a common attachment for the "standard rate codes" file exists in a source region with the key 123456789. The table below highlights possible situations at the target region and actions supported in CMA.

Scenario	Target Situation	Action	Comments
1	No matching record	Record can be added with key 123456789.	
2	Record exists with key 123456789 and logic confirms that it is also the "standard rate codes" attachment.	Record can be updated.	
3	Record exists with key 123456789, but logic detects that it is not the "standard rate codes" attachment.	Record is not updated. An error is issued.	The system cannot update this record because it's not the right attachment record.
4	The system detects that another attachment record exists for the "standard rate codes" attachment with a different ID.	Record is not updated. An error is issued.	Assumption is that the record was created directly in the target or was copied from a different source.

The use cases described in scenarios 3 and 4 above would require key mapping to keep track of the id from the source to the id in the target so that any other records from the source that reference this key as a foreign key would be updated as part of the migration. This functionality is not supported.

Scenarios 1 and 2 above are supported for maintenance objects that use the method to detect the logical key.

NOTE: If a maintenance object with a system generated key does not supply a method to detect the logical key, CMA will update an existing record with the same ID. For maintenance objects in the framework that provide this method, refer to [Framework Provided Migration Configuration](#). For your specific edge application, refer to the CMA addendum for information about support for admin data with system generated keys.

The product recommends that an implementation establishes a migration strategy such that administrative records with system generated keys are always created in the same region and always follow a standard migration path for promoting the data from this source region to other regions. Following this strategy, you would minimize or eliminate the possibility that a record for the same logical key is created in multiple places such that different IDs would be generated as described by scenario 4 above.

MOs with a Mixture of Administration and Non-Administration Data

That there are some MOs that contain a mixture of master or transaction data and administrative data. The Attachment is an example of this. The product supports common attachments and owned attachments. Owned attachments are records that are specific to its owner. The owner could be master or transaction data and its attachments are therefore considered master or transaction data. Owned attachments are not candidates for migration using CMA. Common attachments on the other hand are considered administrative data and may be candidates for migration using CMA. For these use cases, an implementation may follow the suggested strategy of only creating the administrative data in one region so that IDs for

common attachments are not reused. However, it is reasonable and expected that owned attachments are being created in the target region and may receive a system generated key that matches the key of a common attachment from the source region.

To try to minimize this issue, the system includes a special method to be used by any MO that may contain administrative data mixed in with master or transaction data. This method generates the key of an administrative record with a zero (0) in the middle of the key and ensures that the keys for master and transaction data do not include a zero in this spot. For maintenance objects in the framework that use this method, refer to [Framework Provided Migration Configuration](#). For your specific edge application, refer to the CMA addendum for information about additional maintenance objects that may be in this category.

No Record Deletion

The CMA process allows users to define records to copy from a source environment to a target environment. In that way, the import process for the migrated records is able to identify objects to add and objects to change. There is no mechanism for indicating that records in the target environment should be deleted. The absence of those records in the import is not enough because the migration may be only importing a subset to add or update. If data on the target system must be deleted, users must delete the records in the target accordingly.

Note that CMA does support the deletion of child rows of an object as a result of a comparison. This is only applicable to child records that are owned by the implementation.

File Transfer Considerations

When moving the export file between systems, use the binary transfer option of whatever tool you use to move the file so that line-end characters are not converted from Linux-style to Windows-style or vice versa.

It is recommended to avoid using '.txt' for the export file's extension (defined in the [master configuration](#)). That file extension by default implies a non-binary file and tools that perform file transfer may treat this as a non-binary file unless explicitly stated. The recommendation is to define '.cma' as the extension. This is not a recognized file extension and most file transfer tools will transfer the file as is.

Note that if the file gets converted, there are two likely outcomes - either a numeric conversion error, or a buffer under-run error may be received when attempting to import the file.

Multi-Language Environment Considerations

If your implementation uses a language other than English, it means that migrated objects may have multiple language rows (because English is always enabled). There are some important points with respect to multiple languages and CMA:

- As described in [User Language](#), there are steps to follow when supporting an additional language. The steps outlined in that topic highlight that for system data, translation of the strings may be provided via a language pack provided by the product or may be the responsibility of your implementation. In either case, this effort is non-trivial and will have its own established plan. The expectation is that the translation of the system data is applied for each region at your implementation site. CMA should not be used to create a new language in a target region.
- For administrative / control data that your implementation develops as part of your project, the expectation is that descriptions for your supported language are entered in the region that is considered the source region used to promote changes to regions in the "chain". For example, control data is entered in a development region and promoted to a test with the supported language enabled in both regions.
- What if you export data from a region with more languages enabled than your target? This scenario is perhaps a case where the source region is a type of test or playpen region where the additional language is enabled for other purposes. In this case, if the language code does exist at all in the target region, the import will produce an error given that the code is invalid. If the language code exists but is not enabled, this will cause the extra language rows to be inserted in the target region, but will not cause any issues. They are simply ignored.
- What if you export data from a region with fewer languages enabled than your target? In this situation, the import process will only create language rows for the languages that were copied from the source. It will not automatically create language rows in the target as part of the import. For this situation, the recommendation is to run the **New Language** batch program ([F1-LANG](#)) that creates any missing language entries.

CMA Configuration

The following sections describe tasks required for CMA configuration.

Master Configuration - Migration Assistant

Before proceeding with your first migration, system wide configuration must be defined. This is captured in the **Migration Assistant Configuration** [master configuration](#) record. This must be defined in both the source environment and the target environment.

For more information about specific fields in the master configuration, refer to the in-line help.

NOTE: This record can be updated at any time to change details. The new configuration takes effect on all subsequent exports and imports.

Migration Plans

A migration plan defines one or more types of objects that are eligible for migration. It is essentially a set of instructions describing how the data to be exported is structured, allowing objects to be migrated together as a logical unit to ensure consistency and completeness.

The following topics describe defining a migration plan as well as other topics for a migration plan.

Defining a Migration Plan

To view or define a migration plan, navigate using **Admin > Implementation Tools > Migration Plan**.

Use the **Migration Plan Query** portal to search for an existing migration plan. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

CAUTION: Important! If you introduce a new migration plan, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

The following points provide information about defining **Instructions** for a migration plan.

The **Instruction Sequence** uniquely identifies the instruction. The recommendation is to use increments of 10 to allow insertion of other instructions in the future.

Select **Primary** for the first **Instruction Type**. All migration plans must contain one and only one primary instruction. All subsequent instructions require a **Subordinate** instruction type. In this case, the **Parent Instruction Sequence** must be entered. This number, used to maintain the defined relationships in the exported data, must match an instruction sequence number at a higher level in the hierarchy.

The instruction **Description** provides a business description of the instruction.

Select a **Business Object (BO)** to define the type of object from which data will be derived.

NOTE: Though BOs are specified in each instruction, it's important to understand that each BO is used only for filtering purposes. The migrated data set comprises the complete contents of the *maintenance object* that the business object structure is defined against. For a more detailed explanation of this, see [Understanding the BO Filtering Process](#).

NOTE: Refer to [Identifying Tables to Exclude From Migrations](#) for information about defining child tables to always exclude from a migration.

Traversal Criteria is used to define the relationship between each of the objects in a migration plan. The system provides three options to define how the child object is connected to the parent object so the system knows how to traverse from one object to another. **Traversal Criteria Type** options are **Constraint**, **SQL** and **XPath**. The following points explain each option:

- **Constraint** allows you to select a table constraint that represents a given record's relationship to another record in the system via a foreign key constraint defined in the meta-data. If **Constraint** is selected, the following additional fields are enabled:
 - **Constraint ID** is a unique identifier for the constraint. The search will show the valid table constraints for the MO of the instruction's BO and the MO of the parent instruction's BO.
 - **Constraint Owner** is used to define the owner of the constraint. This is populated automatically when selecting a constraint from the search.
- **SQL** lets you specify SQL join criteria between the parent instruction's object and the child object in the **SQL Traversal Criteria**. The syntax of the the traversal criteria is a WHERE clause (without including the word WHERE). When referring to a field on the parent instruction's object, use the syntax `#PARENT.TABLE_NAME.FIELD_NAME`. When referring to a field on the current instruction's object, use the syntax `#THIS.TABLE_NAME.FIELD_NAME`. For example, the following statement is used on a migration plan for Business Object, where the parent instruction is the BO and the subordinate instruction is used to reference the UI Map that is referred to as a BO option with the option type "F1DU":
`#PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_FLG = 'F1DU' AND @trim(#THIS.F1_MAP.MAP_CD) = @trim(#PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_VAL)`.
- The **XPath** option lets you apply syntax in an XPath expression referencing elements in the instructions' referenced business objects. This is entered in the **XPath Traversal Criteria**. For example, the display map collection statement in the SQL example noted above would be written as follows in XPath: `#this/mapCd = #parent/businessObjectOption/businessObjectOptionValue AND #parent/businessObjectOption/businessObjectOptionType = 'F1DU'`. This technique allows foreign key references that are mapped inside a CLOB to be referenced.

NOTE: The `#parent` expressions may access elements that are stored in the CLOB and described using mapXML and mdField. However, the `#this` expressions must refer to fields available in the business object using the mapField reference.

Defining **Next Migration Plan** provides the ability to indicate that in addition to copying the object defined in the instruction, any additional instructions included in that referenced migration plan will also be included in an export.

The **Algorithms** grid contains algorithms associated with each instruction. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the Sequence in which they should execute.

System Event	Optional / Required	Description
Pre-Compare	Optional	Algorithms of this type may be used to adjust the data after it is moved to the target system. These may only be defined on the primary instruction. Refer to Adjusting Imported Data for more information. Click here to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Import	Optional	Algorithms of this type are no longer supported.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [F1_MIGR_PLAN](#).

Understanding the BO Filtering Process

Migration plan instructions require the definition of a business object to provide CMA with information about the record related to the instruction.

If the business object is the physical business object for the maintenance object, then CMA assumes that the instruction applies to all records that satisfy the traversal criteria. CMA recognizes the physical BO by comparing the BO to the value defined in the maintenance object option. If the business object defined is not the physical BO, then CMA will limit the records in the instruction to those that explicitly reference this BO or reference a child of this BO as its [identifying BO](#) value. (In other words, this BO must be in the parentage hierarchy of the records to be included in the instruction.)

NOTE: Unlike Bundling, CMA does not use the BO schema to drive what data is copied for a given record. The BO is only used as a filtering mechanism for selecting records. Refer to [Identifying Tables to Exclude from Migration](#) for information about how to ensure a child table is not included in a migration.

For example, if you define a migration plan for Master Configuration and use the physical business object for the instruction (**F1-MstCfgPhysicalBO**) then all master configuration records are considered for the instruction. If instead the business object you define is Migration Assistant Configuration (**F1-MigrationAssistantConfig**) then only the record related to this business object is included in the instruction.

Migration Plans for Objects with CLOB-Embedded Links

When migrating objects where foreign key references are captured in the object's XML based field, subordinate instructions are needed to define the foreign key references in order for CMA to understand the relationships. This is in contrast to direct foreign keys where CMA can determine the relationships using constraints. The instructions provide two purposes. For wholesale migrations, where all data (or a large amount of data) is being migrated, the instructions allow CMA to group related objects into transactions. This helps in the apply process at import time to ensure that related objects are grouped together. However, the apply process includes iterative steps to try to overcome dependencies like this so defining the instructions is not critical for this type of migration. For piecemeal migrations, defining instructions ensures that the related objects are included in the migration, if appropriate.

The following are options for creating migration plans with XML-embedded links:

- One option is to use the specific logical (business) BO in the primary instruction to define the object you are copying. With this option, the subordinate instructions may use XPath criteria to define the related foreign key. When this approach is used, a separate Migration Plan must be created for each logical BO. (Refer to [Understanding the BO Filtering Process](#) for more information.) This option would only be used in isolated cases.
- Another option is to create a migration plan that uses the Physical BO as the primary instruction, and then include a subordinate instruction for the real logical BO, using SQL Traversal to join the object to itself by its primary key. Note that with this technique, the records that reference the logical BO will still only be included in the export file once. At this point further subordinate instructions may use XPath notation to define the foreign key data. Using the physical BO as the primary instruction ensures that all records in the MO are considered. The subordinate instructions with the logical BO and XPath notations will only apply to the records that are applicable to that BO. This option is useful for MOs that have a small number of logical business objects with disparate foreign keys.
- Another option is to use the physical BO in the primary instruction and use raw SQLs in the subordinate instruction's traversal criteria to identify the foreign keys using substring commands. A separate Subordinate Instruction is needed for each SQL corresponding to each element occurrence. Using this technique has the same advantages of the previous in

that all records for the MO are included in the migration. However, this technique may be useful for maintenance objects with a larger number of business objects expected where each has one or more foreign keys. It's especially useful if many business objects reference the same foreign key. Then only one instruction is required for that foreign key. Note that a single migration plan may use this technique and the XPath technique for different elements.

A migration request may have multiple migration plans for the same maintenance object. That allows for some flexibility and long term maintainability in that the above techniques may be used in multiple migration plans. Consider the following example:

- A product provides base business objects with foreign keys defined in the XML field and provides the appropriate migration plan with instructions. An implementation extends this business object or perhaps creates their own business object for the same maintenance object and includes different additional foreign keys in the XML. Rather than duplicating the base migration plan and adding additional instructions for the additional foreign keys, the implementation can create a second migration plan for the MO with the additional foreign keys defined. A migration request should be defined to include both migration plans. In this case if the implementation has only one custom BO, they can choose to use the custom BO as the primary instruction as described above in the first option.

Defining a Migration Request

Migration Requests are unordered lists of Migration Plans that are to be migrated together. Algorithms, SQL statements, or specific keys are used to specify the set of objects you want to export. When complete, the request describes the complete data set that is extracted to the migration export file when the request is executed.

To view or define a migration request, navigate using **Admin > Implementation Tools > Migration Request**.

Use the **Migration Request Query** portal to search for an existing migration request. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

Define one or more **Migration Plans** for the migration request. Note that the order doesn't matter. During the export process, CMA looks at all objects that qualify for the export and will group them together based on their relationships.

For each option, define an appropriate **Selection Type**, which is used to filter which records in the migration plan's object should be selected. The valid values are **Algorithm**, **SQL Statement**, and **Specific Key**.

- For the option **Algorithm**, specify an **Algorithm** for the Key Selection. The algorithm is responsible for returning a list of key values to include in the migration.

NOTE: The algorithms that are eligible to be plugged in here are those valid for the Migration Request — Select system event. Click [here](#) to see the algorithm types available for this system event.

- For the option **SQL Statement**, specify an **SQL Statement** for the key selection. may be used to limit the keys to include. To migrate all records in the table, the SQL statement "1=1" may be used. Refer to the inline help for other examples.
- For the option **Specific Key**, enter one or more primary keys to individually to define the records to include. Click the "+" to enter each additional key. Each row represents one key and supports a multi-part primary key.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [F1_MIGR_REQ](#).

Wholesale and Piecemeal Migrations

The Configuration Migration Assistant is used for two general types of migrations: wholesale and piecemeal. The following sections provide some additional information about these concepts.

Wholesale Migrations

Wholesale migrations are used when migrating all the configuration and/or administrative data from one environment to another. For example, a wholesale migration might be used when migrating administrative data from a development or test environment to a production environment.

A wholesale migration may be comprised of one or more migration requests that in total include all the administrative data to move. Whether one migration request is used or multiple are used depends on the following considerations:

- For each migration request used, a separate migration data set export record (and therefore separate migration data set import record) is needed. The multiple records may cause more user interaction with the data to progress all the records to their final state.
- On the other hand, splitting data into multiple migration requests, grouping information logically together may allow for more reuse.
- In addition, you should consider that the framework product provides base migration requests and your specific edge product may provide base migration requests as well that may or may not include framework migration plans. Using the product provided migration requests is beneficial with respect to maintenance. As features are added to the product (including new administrative maintenance objects), any impact on CMA should be included in the product owned migration requests. If your implementation introduces new custom administrative maintenance objects that should be included in CMA, then custom migration plans and a custom migration request should be added. Your implementation may choose to include only migration plans for your custom MOs (and simply migrate the objects in this table as a separate step in the process) or include other migration plans from the product in your custom migration plan to have a consolidated plan.

Migration plans used in wholesale migrations may be designed to omit subordinate instructions related to explicit foreign keys that are identified through constraints as they are not needed, assuming that the data they are referring to will also be included in the migration.

NOTE: Refer to [Framework Provided Migration Objects](#) for information about migration requests provided in the framework product. Refer to your specific product's documentation for information about addition base provided migration requests.

Piecemeal Migrations

A piecemeal migration refers to migrating a targeted subset of data from one environment to another. Examples of piecemeal migrations include:

- Migration of a new Business objects with its options and algorithms.
- Migration of a new maintenance portal, its zones, and its application service.
- Migration of all outbound message types.

Administrative users can define a migration request with the specific types of objects that should be copied and use selection criteria to limit the records to copy as desired.

The migration plans included in a 'piecemeal' migration request should be reviewed carefully to verify what subordinate instructions are included. When copying a specific type of record, there may be related records that should be copied as well and other related records that should not be copied. For example, when copying a custom To Do Type, perhaps any algorithms it refers to should be copied (along with their algorithm types) and its message should be copied, but not its navigation option or its drill keys, which would probably refer to base values.

Identifying Tables to Exclude From Migrations

Some maintenance objects that are eligible to be migrated may include child tables that should not be included in the migration. For example, if an object includes log tables, the entries in the log should reflect the actions on the object in that system, and will be different between the source system and the target system. If you have a custom Maintenance Object that includes tables you don't wish to migrate (such as a log table), use the **Non-Migrated Table** option on the MO to specify this table. All child records for this table will also be ignored during migration.

Another use case to consider is a child “many-to-many” table that connects two administrative objects and exists in the maintenance object of both tables. The child table may be in both MOs for convenience sake, but it may be that one MO is considered more of a “driver” object and the other more of a subordinate. If you are doing a targeted (piecemeal) migration where you want to copy a subset of objects, you may want to only copy the driver object and its children and their data but not their children. For example, in Oracle Public Sector Revenue Management, a Form Type includes a collection of valid Form Change Reasons and in turn the Form Change Reason refers to its Form Types. If an implementation wants to do a targeted migration of a form type and include all its related information, including form rules and form change reasons, it does not want the migration of a form change reason to in turn copy all its form types (and their data).

NOTE: The MO option must be set in both the Source and Target systems for a given MO.

Configuring Custom Objects for Migration

During the implementation and extension of the product, new custom administrative maintenance objects may be introduced. If your implementation would like to migrate records in those maintenance objects using the Configuration Migration Assistant, additional steps must be performed, which are highlighted in the following sections.

Physical Business Object

As described in [Understanding the BO Filtering Process](#), the migration plan requires a business object for its instruction. The business object is used to identify the records eligible for inclusion in the migration. Assuming your custom tables use one or more “logical” business objects for their processing, your implementation must decide if these business objects are appropriate for use by the migration plans, or if a physical BO is warranted. If so, create an appropriate physical BO.

Review MO Option Configuration

The following points highlight maintenance object (MO) configuration that should be reviewed or updated to support CMA:

- If a physical BO was created (above), link it to the MO as an option using the appropriate option type.
- Be sure that your MO defines an appropriate [FK Reference](#) and includes an Option on the MO that identifies the FK Reference. This is used by various portals and zones for CMA when showing detail about records being imported into the target region.
- As described in [Identifying Tables to Exclude From Migrations](#), an MO option is used to identify child tables for an MO that should never be included in a migration. If your custom maintenance object includes a standard Log table, than the recommendation is to list that table as an excluded table. Depending on the specific design of the maintenance object, there may be other child tables to define.

Characteristic Type Configuration

The CMA import process will attempt to create a log record for any administrative object that includes a log table. If your implementation has introduced any custom administrative tables that you plan to include in a migration request and it includes a log table, you must, to ensure that the log creation is successful, add your log table as a valid characteristic entity to the characteristic type **F1-MGO** (Migration Object).

Navigate to [Characteristic Type](#) and select the characteristic type **F1-MGO**. Navigate to the **Characteristic Entity** tab and add a row to include the characteristic entity for your custom maintenance object's log table.

Standard CMA Configuration

Create one or more migration plans for the new object, depending on the type of data in the maintenance object and the types of migrations you envision:

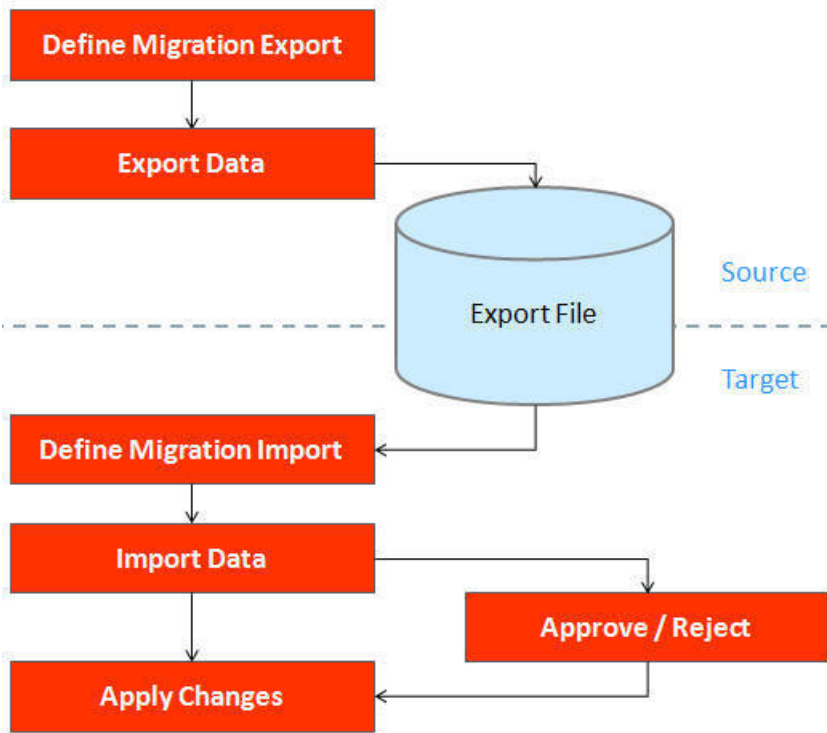
- If you have implemented only one “logical” business object used to define the data in the MO, then a single migration plan that references the this BO (or the maybe the MO’s physical BO) is appropriate.
- If you have implemented more than one “logical” business object, would the data for multiple business objects get copied together? Then perhaps a single migration plan that references the MO’s physical BO is appropriate.
- Are there additional foreign keys defined using mapXML in the business object(s) for the MO? If so, then it is recommended to include sub-instructions to define the links. At this point, if multiple “logical” BOs exist, your implementation may choose to define all the additional elements in the same migration plan or choose to define separate migration plans for each logical BO.
- Your implementation may decide to define more than one migration plan for the same type of record based on the types of migrations you plan to include. For example, you may decide to include a migration plan that copies only the records in this maintenance object. You may decide to define another migration plan that copies the records in this MO along with related records in another MO (for a special type of migration). Having said that, be sure to design the migration plan with reuse in mind to minimize maintenance efforts.

If your implementation has a template migration request to use for piecemeal or wholesale migrations, include the new migration plan(s) as appropriate.

CAUTION: Important! New migration plans and migration requests should follow naming conventions. Refer to [System Data Naming Convention](#) for more information.

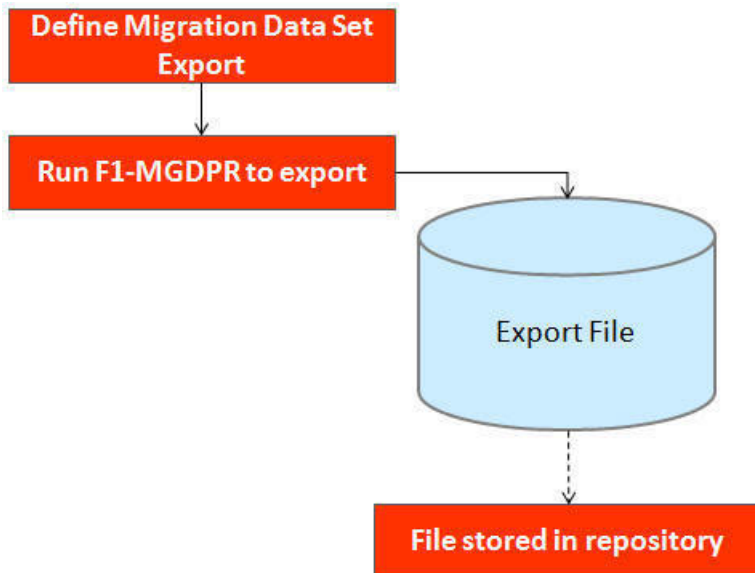
The CMA Execution Process

The following diagram illustrates a high-level view of the Configuration Migration Assistant execution process. The subprocesses illustrated here are described in more detail in the following sections.



Exporting a Migration

The migration export process begins in the source environment by defining a **Migration Data Set Export**, which specifies a defined **Migration Request** and provides a file name and description for the export file. After the data set is defined and saved, the **Migration Data Set Export Monitor** batch job can be submitted generate the export file.



The following topics provide more detail about this process.

Migration Data Set Export

To migrate data from one region to another, define a **Migration Data Set Export**. This establishes the export file name and identifies the migration request that includes the migration plans and selection criteria for the objects to include in the migration.

To view an existing migration data set export, navigate using **Admin > Implementation Tools > Migration Data Set Export**. Use the query criteria to locate the desired data set.

Note that you can also initiate the creation of an export data set from the [Migration Request](#) portal using the **Export** button. The export requires the name of an existing **Migration Request**.

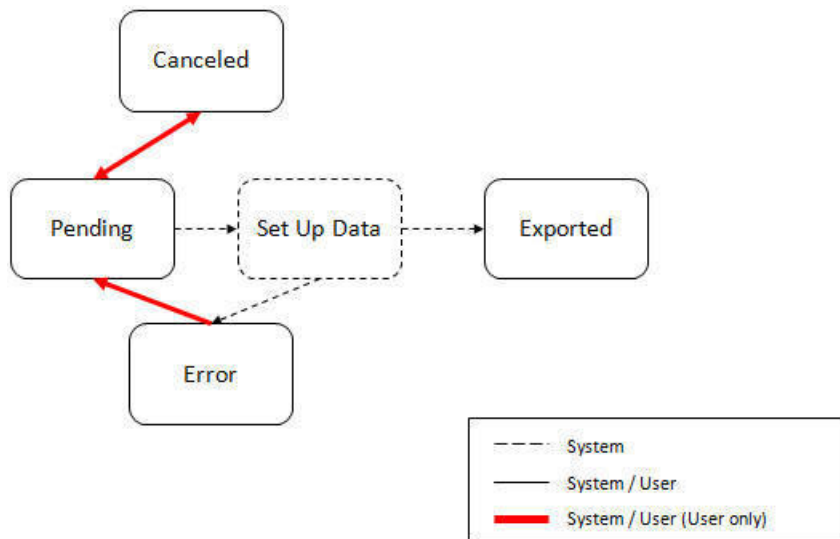
Enter a unique **File Name** for the export. Do not use spaces in the name, and do not enter the file extension or a path. The output location and file extension of the intended export file, which should appear in the **Export Directory** and **File Suffix** labels, are defined in the [Migration Assistant Configuration](#) master configuration record.

Enter an **Export Description** to provide information about the purpose of the export. Note that this field is not language enabled.

The **Source Environment Reference** is for information purposes. It should be populated with text that provides a meaningful description of the source environment. The default value is the URL of the source environment.

Export Lifecycle

The following diagram describes the lifecycle of a Migration Data Set Export (data set).



The following points describe the lifecycle.

- The data set is created in **Pending** status.
- A user may choose to temporarily **Cancel** a pending data set to prevent it from being processed. The user can later return it to the Pending state when desired.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Export Monitor** (F1-MGDPR) selects pending records and transitions them to **Set up data**. Refer to [Running Batch Jobs](#) for more information.
- Set up data is a transitory state that includes the algorithm that does the work of determining the objects to include in the export and group related objects together into a transaction. If everything is successful, the export file is written to the

appropriate file location and the record transitions to **Exported**. If an error is detected, the process stops and the record transitions to **Error**.

- If a record is in error and it is possible to correct the error, the record may be transitioned back to Pending to try again.

When the process is marked as **Exported**, the export file can be imported into the target system.

NOTE: The export process creates a file, providing the benefits of having a standalone file. It can be stored in a version control system for audit purposes or provided to others for import purposes.

CAUTION: Under no circumstances should exported data files be edited manually. Doing so could cause data corruption when the file is applied to the target environment.

NOTE: The export functionality is supported using the business object **Migration Data Set Export** (F1-MigrDataSetExport). The expectation is that implementations will use the delivered base business object and its logic and will have no reason to implement a custom business object for the CMA export process.

Importing and Applying a Migration

The import process is broken down into four general steps: Import, Compare, Approve, Apply. The following points provide an overview of the steps.

- **Import.** The first step covers importing the file and creating appropriate Migration Import records in the target environment to facilitate the subsequent steps.
- **Compare.** The compare step reviews each object that is in the import file and compares the object in the import against the equivalent record in the target environment. The comparison step results in noting which objects are unchanged, which are new (and the appropriate SQL to insert them) and which objects are changed (and the appropriate SQL to update them). Based on user configuration at import time, the objects that qualify for the import may be in a state that requires review or may be pre-approved.
- **Approve.** Once the comparison is complete, the user should review the results. There may be records marked for review. All of these records must be approved or rejected before the import can proceed. When the user is satisfied with the results of the comparison and has completed the review, the import is marked to proceed to the Apply step.
- **Apply.** This is the final step and is the step where the records in the target environment are added or updated. Because of potential high volumes of data and because of possible dependencies between records, this step supports two levels of attempting to apply the records. There is an apply step at the object level and an apply step at the transaction level. This will be described in more detail below.

Import Step

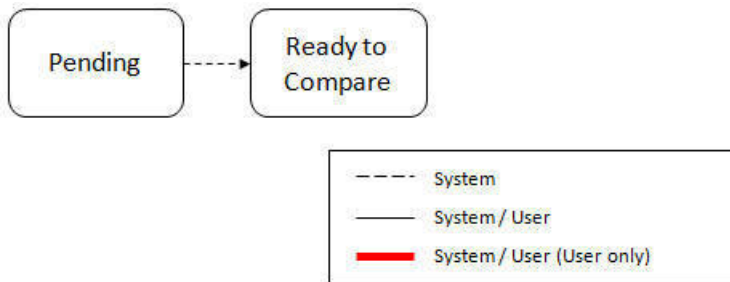
The import process starts with verifying the import directory configured in the [Master Configuration — Migration Assistant](#) for the target environment and ensuring that the exported file is located in that directory. Then, in the target environment, a **Migration Data Set Import** record should be created. The user indicates the file name.

In addition, the user decides what the default status should be for resulting objects.

- The **Default Status for Add** sets the default status for objects that are determined to be *new* during the import comparison process. The default is to automatically set new objects to **Approved** status. Other options are to set any new objects to **Rejected** or **Needs Review** status.
- The **Default Status for Change** sets the default status for objects that are determined to be *changed* during the import comparison process. As with new objects, the default for changed objects is **Approved**, with **Rejected** or **Needs Review** options available.

The file to import contains a list of all the objects included in the export. Any objects that the export step determined to be related have been grouped into “transactions”. Once the Migration Data Set Import is created, the next step is for the system to read in the file and create Migration Transactions and Migration Objects.

The following is a portion of the Migration Data Set Import lifecycle as it pertains to the import step.



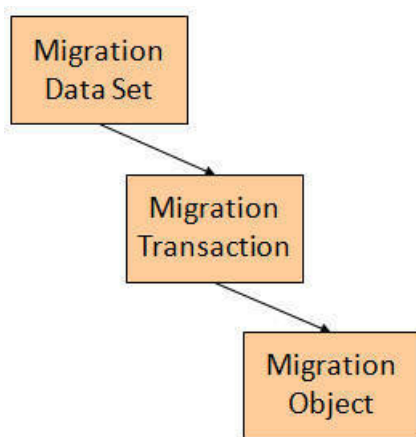
The following points describe the lifecycle.

- The data set is created in **Pending** status.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Import Monitor** (F1-MGDIM) selects pending records and transitions them to **Ready to Compare**. Refer to [Import Process Summary](#) for more information.

The Ready to Compare state has an algorithm that is responsible for reading the related import file and creating the migration transactions and migration objects. The data set remains in this state until the comparison step is complete.

NOTE: A user may choose to **Cancel** a data set. Refer to [Cancelling a Data Set](#) for more information.

The following diagram highlights the relationships of the resulting migration import records.



The migration transaction and migration object each have their own lifecycle that will help manage the subsequent compare and apply steps. At the end of the import step, the status values of the three types of records are as follows:

- Migration Data Set Import is in the **Ready to Compare** state.
- Migration Transaction is in the **Pending** state.
- Migration Object is in the **Pending** state.

NOTE: The import functionality is supported using business objects supplied by the base product. The expectation is that implementations will use the delivered base business objects and their logic and will have no reason to implement a custom business objects for the CMA import process. The base business objects are **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

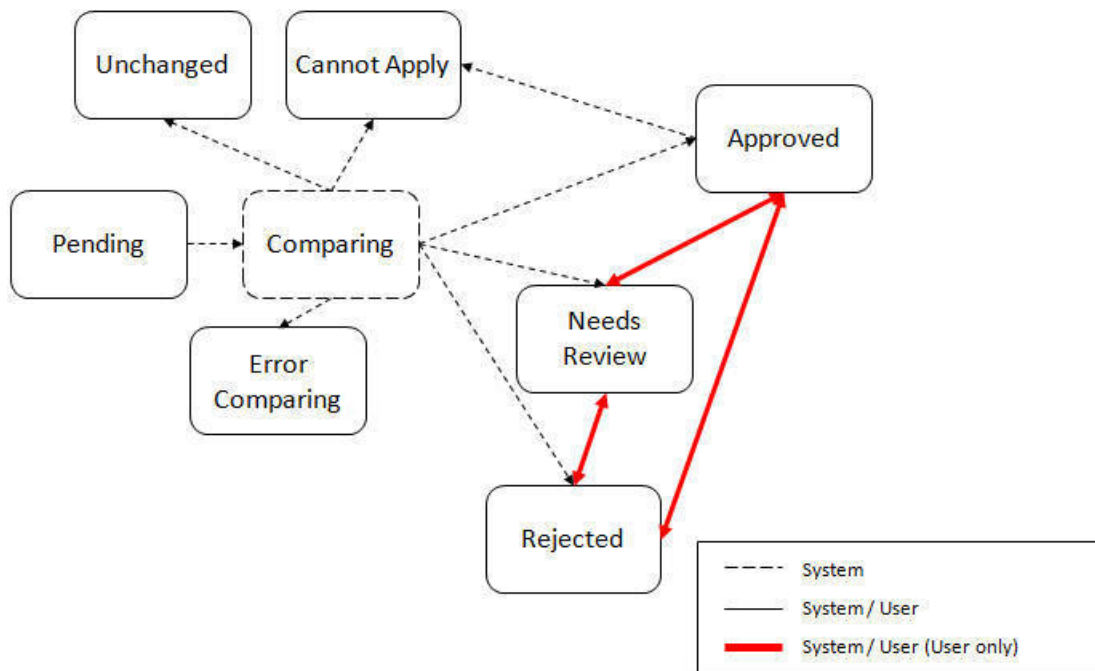
Compare Step

The import step results in the creation of one or more migration objects, one for each record selected in the export based on the export's migration request and its configuration. Related objects are grouped together in migration transactions. The next step in the import process is the Comparison step. In this step, the data captured by the import file for each object is compared to the view of that object in the target environment.

To cater for a possible large volume of objects, the comparison is done via a batch monitor. To aide in performance of the process, the monitor is performed on the migration objects so that it can be run multi-threaded. Once the objects are finished with the comparison, the migration transactions and the migration data set should be updated with an appropriate overall status before continuing to the next step. As a result, the comparison actually requires three steps: Migration Object Comparison, Migration Transaction Status Update and Migration Data Set Export Status Update. The steps are explained in detail in the following sections.

Migration Object Compare

This is the main step of the comparison. The **Migration Object Monitor** (F1-MGOPR) selects pending migration object records and transitions them to **Comparing**. This is a transitory state that includes the algorithm that does the work of comparing. There are various possible outcomes that could occur based on the logic in the algorithm. The following diagram illustrates a portion of the migration object lifecycle that pertains to comparison.



The following points describe the lifecycle.

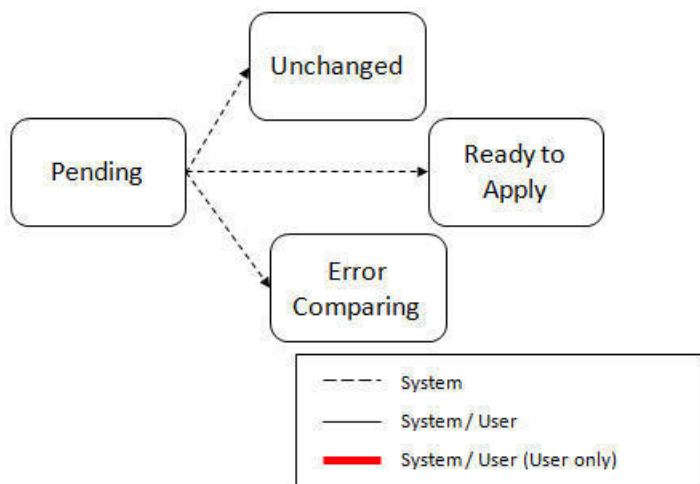
- When **Pending** records are selected by the monitor batch job, it transitions to **Comparing**. If the migration object refers to one or more pre-compare algorithms, they are executed to [adjust the data prior to comparison](#). Then algorithm will determine the appropriate next state by comparing the source data to the target data.
- If the record in the migration object is found in the target environment and the data is exactly the same, the record transitions to **Unchanged** (with the object action value also set to **Unchanged**).
- If the record in the migration object is found in the target environment and the data is different, the algorithm sets the object action value to **Change** and generates the appropriate SQL to be used later in the Apply step to update the record. It then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Change setting captured on the Data Set.

- If the record in the migration object is not found in the target environment, the algorithm sets the object action value to **Add** and generates the appropriate SQL to be used later in the Apply step to insert the record. It then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Add setting captured on the Data Set.
- If there is any issue with attempting to parse the object data from the import, the record transitions to **Error Comparing**.
- If there is any reason that the imported object is not valid for import, the record transitions to **Cannot Apply**. The log will be updated with the error that caused the record to transition to this state. An example is that perhaps the record was exported in a different version of the product and has additional elements that are not recognized in this version.

NOTE: Refer to [Cancelling a Data Set](#) for information about cancelling a data set and its impact on its related objects.

Migration Transaction Status Update

After the import step, the migration transaction remains in the Pending state until all its objects have completed the comparison step. At that point, the status of the transactions should be updated based on the results of their objects. The **Migration Transaction Monitor** (F1-MGTPR) selects pending migration transaction records and runs its monitor algorithms. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates a portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Pending.

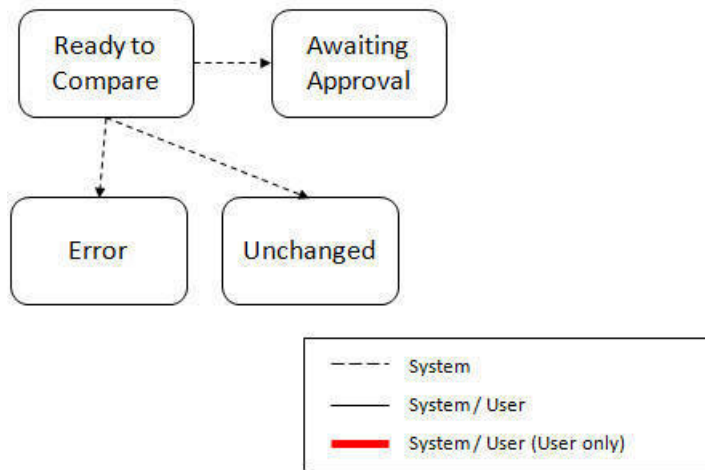
- If any related migration object is in the Error Comparing state, the transaction transitions to **Error Comparing**.
- If all related migration objects are in the Unchanged state, the transaction transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Ready to Apply**. This means that at least one object is in an “apply-able” state.

The transaction remains in the **Ready to Apply** state until a user has approved the data set to move to the Apply step and the transaction’s related objects have attempted to apply themselves. This is described in more detail below.

NOTE: Refer to [Cancelling a Data Set](#) for information about cancelling a data set and its impact on its related objects.

Migration Data Set Import Status Update

Once all the objects and all transactions have been updated via the previous two steps, the migration data set export must be updated based on the results of their transactions. The **Migration Data Set Import Monitor** (F1-MGDIM) selects Ready to Compare data sets and runs its monitor algorithms. Note that this is the same monitor process that is used to select Pending data sets. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates the portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Ready to Compare.

- If any related migration transactions is in the Error Comparing state, the data set transitions to **Error**.
- If all related migration transactions are in the Unchanged state, the data set transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Awaiting Approval**. This means that there are no errors and at least one object is in an “apply-able” state.

The data set remains in the **Awaiting Approval** state until a user decides that the data set and all its records are ready to progress to the Apply step.

NOTE: A user can choose to cancel a data set at any time while it is in progress. Refer to [Cancelling a Data Set](#) for more information.

Approval Step

Once the comparison is complete and the data set transitions to the Awaiting Approval state, a user needs to progress the data set to **Apply Objects** to trigger the Apply step. The following points describe steps a user may take during the approval step.

- If the data set configuration for the Default State for Add and Change was set to **Approved**, then any migration object that is determined to be eligible for the Apply step will be in the Approved state. In this situation, a user may want to review the data set and its transactions and objects to see verify that the results make sense. At that time, the user is able to move an object to Needs Review or Rejected as appropriate.
- If the data set configuration for the Default State for Add and Change was set to **Needs Review** for either option, then each migration object in the Needs Review state must be reviewed and the user must either Reject or Approve each object before moving to the Apply step.
- If the data set configuration for the Default State for Add and Change was set to **Rejected** for either option, the assumption is that the rejected records don’t need to be reviewed. But if a user finds a rejected record that shouldn’t be rejected, it may be transitioned to Approved (or Needs Review) as appropriate.

Once the user is comfortable with the data set’s results and no more objects are in the Needs Review state, the user should transition the record to **Apply Objects**. This will initiate the Apply step.

NOTE: Refer to [Maintaining Import Data](#) for details about the pages provided to help the user review a data set and its transactions and objects to help in the approval step.

NOTE: A user can choose to cancel a data set at any time while it is in progress.

Apply Step

The apply step is the step where records in the target environment are added or updated. Like the comparison step, the apply step is actually multiple steps to optimally handle high volume and dependencies between records as smoothly as possible. Before explaining the apply steps in detail, the following points highlight the type of data that may be included in a given data set.

1. Records that have no foreign keys and therefore no dependencies on other records. Examples: Message, Display Profile.
2. Records that have foreign keys that may already be in the target. Examples: Algorithms for existing algorithm types, To Do Roles for existing To Do Types.
3. Records that have foreign keys that are new records but also part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: Script-based Algorithm Type where the script is also in the migration.
4. Records that have foreign keys that are new records but also part of the migration. CMA did not detect the relationship. This may occur if the reference the foreign key is in a CLOB or parameter column and the migration plan did not include an instruction to explicitly define the relationship. Example, a Zone that references a visibility script.
5. Records that have circular references where both records are new and are part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: plug-in Script for a BO plug-in spot. The script references the BO and the BO references an algorithm for the script's algorithm type.

To handle high volume data, the first step in the apply process is to perform the apply logic at the migration object level via a multi-threaded batch job. This should result in all records in categories 1 and 2 above being applied successfully.

For records in categories 3 and 4 above, if a record with a foreign key is added or updated before its related record, the validation will fail. However, if the related record is added first and then the record referring to it is added, validation will pass. Because the migration objects are not ordered, the multi-threaded batch process may not process records in the desired order. To overcome this potential issue, the Apply step has special functionality, described in detail below.

For records in category 5 above, the circular reference will mean that the apply process at the object level will not successfully add or update these records. The apply process at the transaction level will cover these records. This is described in detail below.

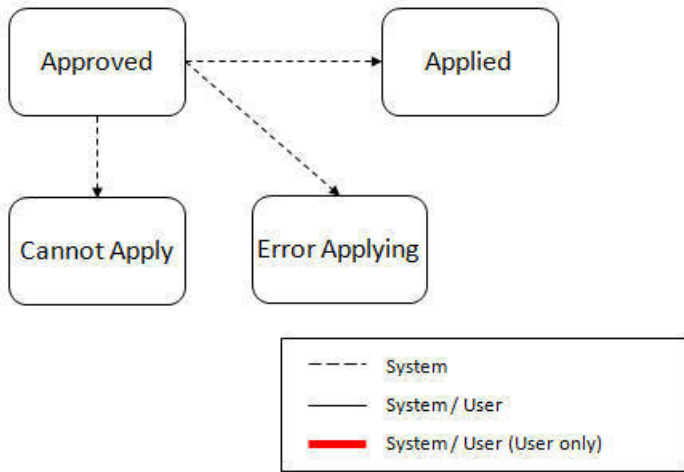
Apply Objects

Once the Data Set is in the state of **Apply Objects**, the **Migration Object Monitor — Apply** process (F1-MGOAP) runs to attempt to apply the objects. The background process in conjunction with the Apply algorithm have special functionality to ensure records in categories 3 and 4 (above) successfully apply during this step:

- The **Migration Object Monitor — Apply** process is a special one that continually re-selects records in the **Approved** state until there are no more eligible records to process.
- When an error is received in the Apply Object algorithm, the algorithm increments an “iteration count” on the migration object record. If the iteration count does not exceed a maximum count (noted in the algorithm), the object remains in the **Approved** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum defined in the algorithm, the record transitions to the **Error Applying** state.

NOTE: When submitting this Apply batch job, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker. Doing this will cause the ‘excess’ threads to wait for the supported number of threads to finish, erasing the benefit of the iteration processing.

The following diagram is the portion of the migration object lifecycle that pertains to the Apply step. Note that this diagram is not complete. A subsequent section provides more information about resolving errors.



At the completion of the Apply monitor process, typically the objects will be in the **Applied** state or the **Error Applying** state. The records in the Error Applying state are in that state for one of two reasons.

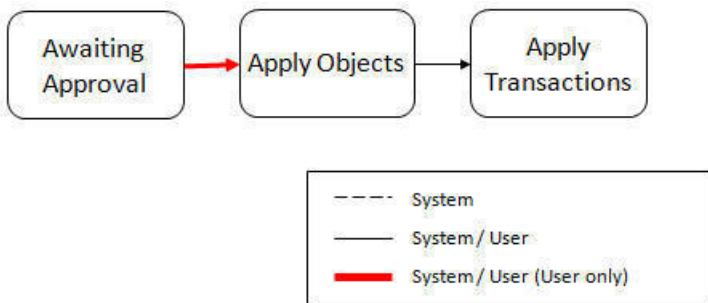
- They are in category 5 described above where the records have a circular reference with another record. For this scenario, the Apply Transactions step described below should successfully apply the records.
- There is some other error that is unrelated to the records in the current migration. In this case, manual intervention is required. Refer to the Resolving Errors section below for more information.

As shown in the diagram, the Apply Objects algorithm may also detect a reason that the object cannot be applied. This may occur if the object in the target environment has been updated since the comparison step, making the SQL captured at that point no longer applicable. If this occurs, after the current migration is fully applied, the original file may be imported again, and new comparisons can be generated and applied.

Apply Transactions

Ideally, after the Apply Objects step, all the objects are **Applied** or are in **Error Applying** due to the “circular reference” situation. The typical next step is to turn over responsibility to the transactions. The migration transactions can then attempt to apply their objects in bulk.

In order to ensure that multiple background processes are not trying to select migration objects to run the Apply step, the transactions are only eligible to attempt to “apply my objects” if the Data Set is in the **Apply Transactions** state.



A monitor algorithm (executed by the data set monitor batch process) on the Apply Objects state checks to see if all migration objects are no longer **Approved** and do not have any records in **Error Applying**. If so, it automatically transitions the record to the **Apply Transactions** state.

If any objects are in **Error Applying**, the monitor algorithm does not automatically transition the record. In that case, a user must decide if the record should transition to **Apply Transactions**. The Resolving Errors section below describes alternative steps that the user may take if there are errors.

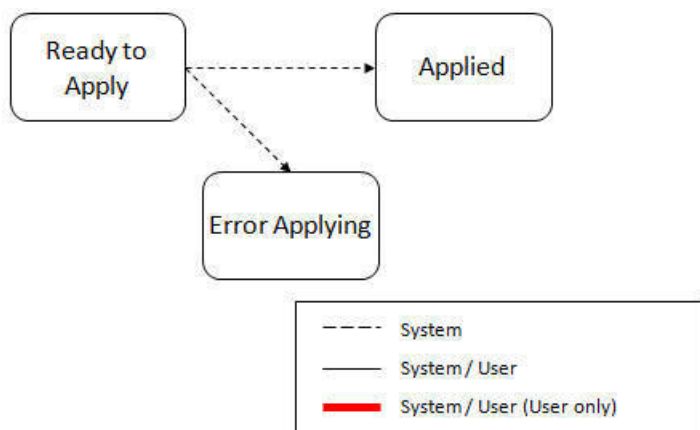
Once the Data Set is in the state of **Apply Transactions**, the **Migration Transaction Monitor — Apply** process (F1–MGTAP) runs. It attempts to apply the transaction’s objects. If no migration objects are in error, the migration transaction

simply transitions to **Applied**. If any of the migration objects are in **Error Applying**, the background process and the Apply algorithm have special functionality to try to overcome dependencies in migrated objects:

- The Apply algorithm selects all migration objects in error and performs all their SQL, then validates all the records. If there are objects in the transaction with circular references, they should pass validation at this point.
- Because there may still be some dependencies across transactions, similar error handling described in the Apply Objects step occurs here. When an error is received in the Apply Transaction's Object algorithm for any of the objects in the transaction, the algorithm increments an "iteration count" on the migration transaction record. If the iteration count does not exceed a maximum count (noted in the algorithm), the transaction remains in the **Ready to Apply** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum, the record transitions to the **Error Applying** state. Note that if any objects in the transaction are in error, none of the objects are applied. They all remain in error.
- The **Migration Transaction Monitor** — **Apply** process is a special one that continually re-selects records in the **Ready to Apply** state until there are no more eligible records to process.

NOTE: When submitting this Apply batch job, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker. Doing this will cause the 'excess' threads to wait for the supported number of threads to finish, erasing the benefit of the iteration processing.

The following diagram is the portion of the migration transaction lifecycle that pertains to the Apply step illustrating the points above. Note that this diagram is not complete. A subsequent section provides more information about resolving errors.



If at the end of the transaction level Apply process there are transactions in error (and therefore there are still objects in error), a user must review the errors and determine how to fix them. Refer to the Resolving Errors section below for more information.

Resolving Errors

As mentioned in the previous sections, errors may be received after the Apply Objects process runs and the system will not automatically transition the data set to the **Apply Transactions**. Rather, a user must make the decision after viewing the errors in the Objects in Error zone on the [Migration Data Set Import](#) portal. This allows for the user to review the errors and make a decision:

- If the errors appear to be dependency related, the user can decide to let the "transactions apply their objects" and transition the data set to **Apply Transactions**, described above.
- If the errors appear to be related to an outside issue that can be manually resolved, the user may choose to fix the issue and redo the Apply Objects step.
- The user may also decide to reject one or more objects to remove them from the migration.

After the Apply Transactions step, if there are still errors, a user must review the records and determine how to proceed. Errors are visible in the Transactions in Error zone on the [Migration Data Set Import](#) portal.

- The user may decide to reject one or more objects to remove them from the migration.
- The user may manually resolve an issue external to the migration and then decide to do one of the following:
 - Redo the **Apply Objects** step. This is recommended if there are a large number of Objects still in error and not a large number of dependencies expected. The benefits of running the Apply Objects multi-threaded will ensure that the process runs efficiently.
 - Redo the **Apply Transactions** step.

Because the objects and transactions are in Error Applying, in order to “retry” the Apply step after manually fixing an error, the system needs to move the records back to the state that allows them to be picked up by the appropriate Apply process. For migration objects, records need to be moved back to **Approved**. For migration transactions, records need to be moved back to **Ready to Apply**. This is accomplished with a Retry Count. This is different from the Iteration count that is used during the apply steps to handle the iterative selection logic. The following points describe the Retry logic for migration objects.

- When the Migration Data Set Import record first enters the **Apply Objects** state, its Object Retry Count is set to 1.
- When Migration Objects first enter the **Approved** state, their Object Retry Count is set to 1.
- When migration objects transition to **Error Applying**, it remains in this state as long as the object’s Retry Count matches the data set’s Object Retry Count (or it is successfully applied via the Apply Transactions step).
- If a user decides to **Retry Objects** (using an action button on the Migration Data Set Import page), the data set’s Object Retry Count is incremented and the Data set returns to the **Apply Objects** state.
- At this point, the monitor on the **Error Applying** state for the objects detects that the object’s Retry Count doesn’t match the data set’s Object Retry Count and that triggers the transition back to **Approved**. This increments the object’s retry count to match the data set’s. Now the objects are eligible to be picked up by the object level Apply process.

Analogous logic exists for the migration transactions.

- When the Migration Data Set Import record first enters the **Apply Transactions** state, its Transaction Retry Count is set to 1.
- When Migration Transactions first enter the **Ready to Apply** state, their Transaction Retry Count is set to 1.
- When migration transactions transition to **Error Applying**, it remains in this state as long as the transaction’s Retry Count matches the data set’s Transaction Retry Count.
- If a user decides to **Retry Transactions** (using an action button on the Migration Data Set Import page), the data set’s Transaction Retry Count is incremented and the Data Set returns to the **Apply Transactions** state.
- At this point, the monitor on the **Error Applying** state for the transactions detects that the transaction’s Retry Count doesn’t match the data set’s Transaction Retry Count and that triggers the transition back to **Ready to Apply**. This increments the transaction’s retry count to match the data set’s. Now the transactions are eligible to be picked up by the transaction level Apply process.

Finalize Apply Step

Once all the migration objects for a migration transaction are in a final state (**Applied**, **Rejected** or **Cannot Apply**), the migration transaction transitions to the **Applied** state. Once all the migration transactions are in the **Applied** state, the Migration Data Set record transitions to the **Applied** and the import is complete.

NOTE: To review the full lifecycle for each record, refer to the Business Object — Summary tab in the application metadata for the base business objects **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

Adjusting Data Prior to Comparing

Some records may have data that is specific to the environment it is in and won't apply in the target environment. In such cases, an algorithm plugged into the [migration plan](#) primary instruction may be used to adjust the data when importing. This algorithm is executed by the comparison algorithm before any comparison is performed. Algorithms of this system event receive the view of the source record (being imported) and the view of the existing record in the target region, if it exists. The data is provided using the physical BO of the migration plan's maintenance object. The algorithm may make changes and pass a new view of the record that should be used for the comparison. This system event supports multiple algorithms that are executed in sequence. Each algorithm receives the original record's data, the target record's data (if applicable) and the 'new' view of the data (as populated by previous algorithms, if any). The final 'new' view of the data is used for the object comparison.

FASTPATH: Refer to [Base Business Objects](#) for more information about physical BOs.

Some examples of records that may require import algorithms.

- Batch Control references its next batch sequence number along with snapshot information like the last run date / time. This information is only relevant with respect to its environment. The instruction for a batch control can include an algorithm to not overwrite the batch sequence number when copying a batch control.
- Some products include administrative objects that reference a master data object. Master data objects are not copied as part of CMA. An import algorithm may be used to adjust the referenced master data foreign key when importing, for example to reset it (or not overwrite when updating). If the algorithm knows how to find the appropriate master data record to link, that may also be included.

Note that it is possible to use the algorithm to "reset" the source data as a way of indicating that the record should not be imported. For these situations, the migration object comparison step will transition the record to **Unchanged** and will use an object action value of **Canceled**. (Note that object action is a simple lookup value. The record is not transitioned to the **Canceled** BO state as to reserve that status for user initiated cancellations of the object or one of its parent records). This technique not expected to be used often because ideally using appropriate selection criteria at export time should ensure that the only records exported are those that should be imported.

NOTE: Legacy 'Import' system event. The system originally provided an Import system event / plug-in spot. The purpose of algorithms for this plug-in spot were similar in that they were meant to adjust imported data prior to adding or updating. The algorithms were executed in the Apply step. The logic does not allow for easily interacting with the record using a BO. This makes it difficult to use a plug-in script as the plug-in type. In addition, it is difficult to update elements in an XML column. The support for the plug-in spot will be removed in a future release. Algorithms to adjust the data should be using the pre-compare system event.

Import Process Summary

The following table summarizes the steps required to complete the import process from start to finish. Note that this section **only a summary** and assumes that you are familiar with the details described in the previous sections. It highlights what steps are manual and what steps are performed by a batch monitor process. For each step, the table highlights the Next Action sequence that would occur. For the Apply steps, there are two parts where multiple next actions are possible based on whether there are errors and the user's decision on how to resolve the error. Refer to [Resolving Errors](#) for more information. The possible next actions have the same sequence with a letter following the sequence highlighting the action to take based on the results of the previous step.

NOTE: When running the Apply batch jobs, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.

Also note that a sequence and action marked in bold is considered the "normal path".

Step	Seq	Action	Manual / Batch	Portal — Action	Batch Control	Record Impacted — Resulting Status	Next Action Sequence
Import	10	Create Import record	Manual	Migration Data Set Import - Add		Migration Data Set Import - Pending	11
Import	11	Import file	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Ready to Compare Migration Transaction - Pending Migration Object - Pending	20
Compare	20	Migration Object Compare	Batch		F1-MGOPR - Migration Object Monitor	Migration Object - Approved, Needs Review, Rejected, Unchanged or Error Comparing	21
Compare	21	Migration Transaction status update	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply, Unchanged or Error Comparing	22
Compare	22	Migration Data Set Import status update	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Awaiting Approval, Unchanged or Error	30
Approval	30	Review comparison results, approve / reject as needed	Manual	Migration Data Set Import, drill in to the Transactions / Objects as necessary.		Migration Object - Approved or Rejected (no records should be in Needs Review) Migration Data Set Import - Apply Objects	40
Apply	40	Apply Objects	Batch		F1-MGOAP - Migration Object Monitor - Apply	Migration Object - Applied or Error Applying	41 Appropriate next action is based on error review, if applicable.
Apply	41a	Migration Data Set Import status update - auto transition to Apply Transactions. Only applicable if no migration objects are in Error Applying	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42
Apply	41b	Migration Data Set Import status update	Manual	Migration Data Set Import -		Migration Data Set	42

Step	Seq	Action	Manual / Batch	Portal — Action	Batch Control	Record Impacted — Resulting Status	Next Action Sequence
		<p>- manual transition to Apply Transactions.</p> <p>Occurs if user reviews errors and determines that they may be resolved in the Apply Transaction step.</p>			click Apply Transactions	Import - Apply Transactions	
Apply	41c	<p>Migration Data Set Import status update - manual transition to Retry Objects.</p> <p>Occurs if user reviews errors and decides to fix an external error and wants to try the batch level Apply again at the object level.</p>	Manual		Migration Data Set Import - click Retry Objects	Migration Data Set Import - Apply Objects	44
Apply	42	<p>Apply Transactions</p>	Batch		F1-MGTAP - Migration Transaction Monitor - Apply	<p>Migration Transaction - Applied or Error Applying</p> <p>Migration Object - Applied or Error Applying</p>	43 Appropriate next action is based on error review, if applicable.
Apply	43a	<p>Migration Data Set Import status update - auto transition to Applied</p> <p>Applicable if all transactions are Applied</p>	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Applied.	N/a
Apply	43b	<p>Migration Data Set Import status update - manual transition to Retry Objects.</p> <p>Occurs if user reviews errors and decides to fix an external error and wants to try the batch level Apply again at the Object level.</p>	Manual		Migration Data Set Import - click Retry Objects	Migration Data Set Import - Apply Objects	44
Apply	43c	<p>Migration Data Set Import status update - manual</p>	Manual		Migration Data Set Import - click Retry Transactions	Migration Data Set Import - Apply Transactions	45

Step	Seq	Action	Manual / Batch	Portal — Action Batch Control	Record Impacted — Resulting Status	Next Action Sequence
		<p>transition to Retry Transactions.</p> <p>Occurs if user reviews errors and decides to fix an external error and wants to try the batch level Apply again at the Transaction level.</p>				
Apply	44	<p>Update Objects from Error Applying back to Approved.</p> <p>Occurs if user chose to Retry Objects.</p>	Batch	F1-MGOPR - Migration Object Monitor	Migration Object - Approved	40
Apply	45	<p>Update Transactions from Error Applying back to Ready to Apply.</p> <p>Occurs if user chose to Retry Transactions.</p>	Batch	F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply	42

The following table summarizes the batch monitor jobs that are used in the import process. You can see that there are special monitor processes for the Apply step for both the Object and Transaction. However, for all other states that have monitor logic, the standard monitor process for that MO is used.

Batch Control	Description	Comments
F1-MGDIM	Migration Data Set Import Monitor	<p>Processes data set records in the following states:</p> <ul style="list-style-type: none"> • Pending • Ready to Compare • Apply Objects • Apply Transactions
F1-MGTPR	Migration Transaction Monitor (Deferred)	<p>Processes transaction records in the following states:</p> <ul style="list-style-type: none"> • Pending • Error Applying
F1-MGTAP	Migration Transaction Monitor - Apply	<p>Processes transaction records in the Ready to Apply state where the data set is in the Apply Transactions or Canceled state.</p> <p>NOTE: Be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.</p>
F1-MGOPR	Migration Object Monitor	<p>Processes object records in the following states:</p> <ul style="list-style-type: none"> • Pending

Batch Control	Description	Comments
F1-MGOAP	Migration Object Monitor - Apply	<ul style="list-style-type: none"> • Needs Review (check for Data Set cancellation) • Rejected (check for Data Set cancellation) • Error Applying <hr/> Processes object records in the Approved state where the data set is in the Apply Objects or Canceled state. NOTE: Be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.

Refer to [Running Batch Jobs](#) for more information about managing the batch jobs.

Canceling a Data Set

A user may choose to **Cancel** a data set to prevent it from being processed at any point during the process.

If related migration transactions or migration objects have already been created, they will not be canceled as part of the data set getting canceled (due to possible high volumes of related records). They will be canceled the next time an appropriate monitor batch process runs. The child records checks to see if the data set has been canceled prior to any state transition.

Additional Note Regarding Imports

The following points describe miscellaneous comments related to Migration Import.

- CMA relies on the fact that database referential integrity constraints are not in place, and that the SQL statements can be run in any order within the transaction. Any archiving solution that requires referential integrity constraints (such as Information Lifecycle Management) would not be possible on this data. Given that CMA migrations comprise administrative data and not transactional data, this should be a reasonable exception.
- The validation that is performed is only via the **Page Validate** service. BO validation algorithms are not executed. Page validation does not include validation of the business object against the schema (for example, for required fields, field sizes, etc.).
- All migration requests can be exported at the same time. On the import side, however, you should consider importing, reviewing, and applying an entire file/data set before moving on to the next one. The reason is that if objects are included in more than one file, two sets of "inserts" will be generated, but only the first will succeed. The second will cause an insert error on the object, and the transaction would be marked with status "Error Applying". If instead you wait until the first file is applied before importing the second, the second data set will not generate any SQL for the object, since it has already been inserted. It's a matter of efficiency: If you first import all files and then try to apply all, you'll have to identify the duplicated object as an error and then mark the object as rejected before applying the transaction.

Caching Considerations

Because CMA updates administrative data that is usually read from a cache, after a successful migration, the target region now has new administrative data which needs to be part of various caches. It is recommended to flush the server cache (which will trigger a 'global' flush of the cache). If the thread pool workers in the target region are configured to refresh their caches when a global flush is requested, then this is the only step required. If not, then the **F1-FLUSH** batch job should also be submitted to refresh the caches used in batch processing.

FASTPATH: Refer to [Caching Overview](#) for more information.

Maintaining Import Data

This section describes the portals provided to add, view and maintain migration import data.

Migration Data Set Import

Use the Migration Data Set Import portal to view and maintain migration data set import records. Refer to [Importing and Applying a Migration](#) for an overview of the import process.

Navigate using **Admin > Implementation Tools > Migration Data Set Import**. You are brought to a query portal with options for searching for import data sets. In addition, the query provides an option to specifically search for data sets that have either objects in error or transactions in error.

Once a data set has been selected, you are brought to the maintenance portal to view and maintain the selected record. The following zones are visible on the main tab:

- **Migration Data Set Import.** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Migration Data Set Transactions.** This zone is visible once the [Import Step](#) has occurred and lists all the transactions that are related to the data set. To see more information about a specific migration transaction, click the hypertext for its ID. This brings you to the [Migration Transaction](#) portal.
- **Migration Data Set Impacted Object Summary.** This zone is visible once the [Import Step](#) has occurred and lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the [Migration Object](#) portal.
- **Migration Data Set Objects in Error.** This zone is only visible if the data set's status is **Apply Objects**, and there are objects in the status **Error Applying**. It indicates the error for each object. A user may use this zone to review errors after the monitor batch job to apply objects completes. Using the error information shown, the user can choose to transition the record to **Error Applying** or manually fix the cause of the error and click **Retry Objects**.

NOTE: Refer to [Apply Step](#) for more information about resolving errors.

- **Migration Data Set Transactions in Error.** This zone is only visible if the data set's status is **Apply Transactions**, and there are transactions in the status **Error Applying**. It indicates the error for each object. A user may use this zone to review errors after the monitor batch job to apply transactions completes. The errors received when attempting to apply objects at the transaction level may differ from those received when attempting to apply objects at the object level. A transaction log is created for each object error received and these exceptions are shown in this zone.

NOTE: Refer to [Apply Step](#) for more information about resolving errors.

Migration Transaction Portal

This page appears after drilling into a specific migration transaction from the migration data set portal or from the migration object portal.

Refer to [Importing and Applying a Migration](#) for an overview of the import process.

The following zones are visible on the main tab:

- **Migration Transaction.** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Migration Transaction Objects.** This zone lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the [Migration Object](#) portal.

Migration Object Portal

This page appears after drilling into a specific migration object from the migration data set portal or from the migration transaction portal.

Refer to [Importing and Applying a Migration](#) for an overview of the import process.

The **Migration Object** zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.

Running Batch Jobs

There are several batch jobs that are part of the CMA process, especially the import step (which are highlighted in [Import Process Summary](#)). And in some cases, a single batch jobs may process multiple states in the same business object lifecycle. Implementations must decide the best way to manage the batch job submission depending on how they plan to work.

- **Batch scheduler.** If an implementation wishes to put these batch jobs in the batch scheduler, a given job may need to be included several times to manage progressing the records to completion.
- **Timed Batches.** The batch controls can be configured as timed batches so that they run every N minutes based on the setting. This allows for the batch jobs to run periodically and process whatever is ready. A user doesn't have to manually submit a batch request. Navigate to the [Batch Control](#) page and select the appropriate batch controls. For each one, change the Batch Control Type to **Timed**. Fill in the additional information that appears for timed batches.
- **Event Driven.** The system provides BO enter plug-in algorithms that automatically submit the appropriate next batch job for that step in the process. This is not applicable for every step that requires a batch job to be submitted because in some cases the user must make a decision first. Note that configuration is required because the BOs are not configured for this scenario by default. The following highlights the BO and status where an algorithm may be plugged in and the name of the algorithm to use.
 - Migration Data Set Export (**F1-MigrDataSetExport**), **Pending** status: algorithm **F1-MGDPR-SJ** (Submit Migration Data Set Export Monitor).
 - Migration Data Set Import (**F1-MigrDataSetImport**), **Pending** status: algorithm **F1-MGDIM-SJ** (Submit Migration Data Set Import Monitor).
 - Migration Data Set Import (**F1-MigrDataSetImport**), **Ready To Compare** status: algorithm **F1-MGOPR-SJ** (Submit Migration Object Monitor).
- **Manual submission.** The user managing the CMA import process submits the appropriate batch jobs on demand when a particular step is ready. Navigate to [Batch Job Submission](#), select the appropriate batch control and fill in the parameters as needed.

Note that after successfully applying the migrated data, the L2 cache of all thread pools must be refreshed. For more information, see [Caching Considerations](#).

CAUTION: Be sure that the Thread Count set when submitting the batch job does not exceed the number supported by the thread pool. Otherwise the extra threads will simply wait until the supported number of threads are finished.

Refer to the parameter descriptions in the batch control metadata for more information about filling in the parameters.

For additional details on submission controls, refer to the topic [Batch Job Submission - Main](#) in the Batch Jobs section.

CMA Reference

This section provides additional reference information.

Framework-Provided Migration Configuration

This topic describes special information relating to migration objects provided for use by CMA in the product. Additional objects may be provided by your specific product. Any special information for objects is provided separately in each product's documentation.

You can see all currently available objects and their descriptions by performing a search in the **Migration Plan Query** or **Migration Request Query** pages.

The following points highlight some information about the Framework-provided migration plans and migration requests:

- Fields and characteristic types are not migrated with the object unless specifically indicated.
- The **Application Service** used by an object is migrated only if it is CM-owned.
- The migration request **F1-SchemaAdmin** defines migration plans related to data that is commonly used in schema-based objects.
- The **Batch Control** object optionally references a User. If this user does not exist on the target system, CMA cannot apply the requested changes. Also note that when running a batch job, snapshot information is captured on the batch control. Updates like this increment the version number. If a batch control record is part of the migration and the comparison step has detected a change to the batch control, the Apply step will error out for this batch control if a batch job is submitted between the compare and apply step.

NOTE: CMA batch controls that are part of the import step are executing and as such, it is not recommended to include those batch controls in a migration. If your implementation changes default parameters for any of the batch controls, the recommendation is to manually make those changes to the target region.

- There are some system data objects where no information in a base delivered record may be modified by an implementation. For these records, the base delivered migration requests include selection criteria to only select CM-owned records (because the base records will always exist in the target region assuming both regions have the same release). An example is Algorithm Type. The **F1-FrameworkAdmin** migration request only includes CM-owned algorithm types. However, many system data objects support custom changes to one or more fields, for example the Zone object allows an implementation to override the zone text or certain parameters. Other system data objects support custom additions to a collection. For example, the Maintenance Object allows an implementation to add algorithms or options. For the migration plans related to these system data objects, all records are included in the base delivered migration requests to allow for any customized configuration to be migrated. It means that during the Import / Compare step many base delivered objects that are not customized will be marked **Unchanged**.
- The base migration plans for MO and BO include instructions to copy option types that use foreign key references to refer to other objects. Note that the data stored in the options are not validated, so defining these instructions is not required when doing wholesale migrations. However, including subordinate instructions for foreign key references is useful for piecemeal migrations to ensure that the related data is included in the migration. If you add additional MO or BO option types that use foreign keys and you want to support piecemeal migrations, you must create custom migration plans and requests for MO and BO, respectively to include these referenced objects in the migration plan. Note that you do not need to duplicate the instructions in the base migration plans. You may define the additional migration plans to only have the additional custom option types. When submitting a migration request for MO or BO you must include both the base migration plans and the custom migration plans in the request.
- The migration plans provided by CMA migrate scripts, schema-based objects and zones, and, through constraints, some of the associated data associated with them. However, data specified through alternate formats (such as through **Edit Data** steps in scripts, referenced in schemas for schema-based objects, or data from mnemonics in zone parameters, etc.) are not identified and combined in the same transaction. It's important to note that this could cause validation errors during **Apply**, and may require retrying or migrating the additional data separately. See [Apply Step](#) for a description of how to reapply the import should an **Applied With Errors** result occur.

- There are two migration plans for **Scripts**. The migration plan **F1-ScriptOnly** migrates just the script and its **Application Service** (provided the Application Service is CM-owned). This is used in the **F1-FrameworkAdmin** migration request, which migrates all admin objects. The migration plan **F1-Script** includes most related objects, but does not migrate any objects referenced in the edit data area steps. It does not move the **Function** maintenance object (which has been deprecated). This migration plan is not included in any base migration requests. It may be included in any appropriate custom piecemeal migration request where scripts and related data should be migrated.
- If your implementation includes a **Feature Configuration** setting for the **F1_DBCONINFO** entry that will be included in a migration request, be sure that the import user on the target region has the appropriate security rights to this entry (**Super User** access mode for the Feature Configuration application service (**CILTWSDP**)).
- The common attachments in the Attachment maintenance object may be considered administrative data to include in a migration. Because this MO has a system generated key, as described in [Migration Assumptions, Restrictions and Recommendations](#), it uses a logical key of the file name and the creation date to determine if the record exists in the target environment. In addition, this MO contains admin data (common attachments) and non-admin data (owned attachments). To try to minimize the possibility of key “collision”, new common attachments receive a generated key that includes a zero in the middle whereas owned attachments receive a generated key that does not have a zero in the middle.
- The Menu maintenance object has a user defined key, however, its menu lines and menu items have system generated keys. To avoid the possibility of overriding a menu line or menu item incorrectly, the menu MO will check the menu line’s menu name in the source and target to be sure they match and will check the menu item’s menu line in the source and target to be sure they match otherwise an error will be issued in the comparison step.
- Many of the provided Framework-owned migration requests include the SQL statement "1=1" so that all records in the plan are migrated. If an implementation wants to limit a migration to a subset of records, the prepackaged migration request may be duplicated and then customized to modify the key selection criteria.
- Many of the XAI-related maintenance objects include references to environment-specific data, and data should be migrated with extreme care. Migration plans are not provided for MOs that contain mostly environment-specific data. Migration plans are provided for the following XAI-related maintenance objects, but these plans are not included in the **F1-XAI-Admin** migration request: **External System**, **XAI Receiver**, **Message Sender**, and **XAI Route Type**.

Migration Requests for Wholesale Migrations

The following framework provided migration requests may be used in a wholesale migration. Refer to your specific product’s CMA documentation for its recommendation on which migration requests to use for a full migration of framework and product administrative tables.

- The **F1-SchemaAdmin**(Framework Schema Admin) migration request. This request contains migration plans for objects needed by schema based objects, such as Field and Lookup. It has few dependencies on its data and many other objects include dependencies to it.
- The **F1-FrameworkAdmin** (Framework Admin) migration request. This includes migration plans for most widely used maintenance objects Your product may have chosen to incorporate migration plans from this migration request directly into one of its owned migration requests.
- The **F1-GeneralSystemOptions** (General System Objects) migration request. This includes migration plans for business related objects that may or may not be applicable to all products. Your product may have chosen to incorporate a subset of the migration plans from this migration request directly into one of its owned migration requests.

For each migration request determined to be needed for a wholesale migration, create a migration data set export record and process it as documented in **Exporting a Migration**.

Executing Piecemeal Migrations

Piecemeal migrations involve copying a targeted set of data. It may be that it includes a subset of maintenance objects (for example all Activity Types or all Asset Types) or it may further limit the records within one or more maintenance objects (for example, all ‘electric’ rates). Many piecemeal migrations are ad-hoc, depending on the specific requirements at the time.

If your product has provided migration requests for piecemeal migrations out of the box, it would they would be defined to copy the appropriate type of data. However, the selection criteria would most likely not be limited to a type of record. As such, it could be used as is for a “copy all data in these maintenance objects” migration. For example, the framework provides a migration request for Security Configuration. This may be used to copy all the data in the various security tables. Oracle Public Sector Revenue Management provides a migration request to copy form types and its related data.

If your implementation wants to copy only a subset of data for maintenance objects covered by a base provided migration request, do the following:

1. Duplicate the base migration request.
2. Find the migration plan for the object or objects where you want to limit the data. Enter appropriate Key Selection criteria to select the desired data.
3. Use that migration request to request a migration data set export request.

If your piecemeal migration requirement is not satisfied by a base migration request that may be used as a starting point, review the base migration plans for the records that should be copied. Determine if the migration plans have the desired subordinate instructions to copy (or not copy) the related data as desired.

- If existing migration plans satisfy your requirements, create a migration request that includes those plans (and desired key selection as needed). Use that migration request to create a migration data set export request.
- If existing migration plans do not satisfy your requirements, create migration plans as needed. Create a migration request that includes those plans (and desired key selection as needed). Use that migration request to create a migration data set export request.

Configuring Facts

Fact is an optional configuration tool for simple workflow-type business messages or tasks. The base package does not provide a dedicated Fact user interface because fact is generic by design. Implementations configure their own user interface to visualize the desired custom business process. The topics in this section describe the generic Fact entity and how it can be customized.

Fact Is A Generic Entity

The Fact maintenance object is a generic entity that can be configured to represent custom entities and support automated workflows for a variety of applications. Each fact references a business object to describe the type of entity it is. A status column on the fact may be used to capture its current state in the processing lifecycle controlled by its business object.

The maintenance object also supports a standard characteristic collection as well as a CLOB element to capture additional information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI_FACT](#)

Fact's Business Object Controls Everything

A fact's business object controls its contents, lifecycle and various other business rules:

- Its schema defines where each piece of information resides on the physical Fact maintenance object.
- It may define a lifecycle for all fact instances of this type to follow. Each fact must exist in a valid state as per its business object's lifecycle definition.
- It may define validation and other business rules to control the behavior of facts of this type.

FASTPATH: For more information about business objects, refer to [The Big Picture of Business Objects](#).

Fact Supports A Log

The Fact maintenance object supports a log. Any significant event related to a Fact may be recorded on its log. The system automatically records a log record when the fact is created and when it transitions into a new state. In addition, any custom process or manual user activity can add log entries.

FASTPATH:

Refer to [State Transitions Are Audited](#) for more information on logging. For more information about the various configuration tools available, refer to [Configuration Tools](#). For more information about user interfaces, refer to [Portal and Zone Features](#).

Customer Care and Billing Administrative User Guide

Defining General Options

This section describes control tables that are used throughout Oracle Utilities Customer Care and Billing.

Defining Installation Options

The topics in this section describe the various installation options that control various aspects of the system that are specific to the Oracle Utilities Customer Care and Billing product.

FASTPATH:

Refer to [Installation Options - Framework](#) for options that are common to products on the same framework.

Installation Options - Main

Select **Admin > General > Installation Options** and use the **Main** page to define system-wide installation options.

Description of Page

Use **Quick Add Tender Type** to define the tender type [defaulted on payments added using the Payment Quick Add transaction](#).

Use **Starting Balance Tender Type** to define the tender type of the starting balance recorded on your tender controls (this will almost always be the tender type associated with "cash"). This value is used during tender control balancing as a separate balance is required for each tender type in order to balance a tender control. Refer to [The Lifecycle Of A Tender Control](#) for more information.

FASTPATH:

For more information, refer to [Setting Up Tender Types](#).

Turn on the **Create Field Activity Start Stop** if field activities should be created when a start or stop is recorded (as opposed to shortly before the start / stop date). You might want to turn this switch off if it's possible for the state of the service point (or its meter / item) to change between the time service is requested and the actual service date. Why? Because the state of the service point and the state of its meter / item affects the type of field activity that is created. For example, if a customer wants to start service and there is no meter at the metered service point, an "install meter" field activity is created. However, if by the time the install date comes around, a meter has been installed by some other means; this field activity is inappropriate. This is why you might want to setup the system to wait until shortly before the service date to create the field activity (i.e., it reduces the likelihood that an inappropriate field activity is created). Refer to [Starting Service and Field Activities](#) for more information.

NOTE:

Appointments require field activities. If you don't create field activities when service is started / stopped, you cannot use the appointment scheduling functions. Refer to [The Big Picture of Appointments](#) for more information.

If you use orders to create new customers, define the **Campaign** that should be defaulted on orders created when the order transaction is opened for a new customer. Refer to [Real time Marketing of Services to a New Customer](#) for more information.

Use the **Premise Geo Type** to indicate whether at least one geographic identifier (e.g., GPS coordinate) is Required or Optional on a premise. Refer to [Defining Geographic Types](#) for more information.

The **Alternate Representation** flag should be set to None unless your organization uses multiple character sets for a person's main name and / or a premise's address. Alternate representations are typically only used in countries where multiple character sets are used. For example,

- In Hong Kong, a person's name may be written in both Chinese characters and in English.
- In Japan, a person's name may be written in both Kanji and Katakana.

In both of the above situations, users need to be able to use both representations to find a customer or a premise.

NOTE:

Spouses. If your organization doesn't use multiple character sets, you might want to consider using this functionality for spousal relationships. For example, rather than setup a person for each party in a spousal relationship, you could simply define one party using the person's main name and the spouse using the alternate name. While this is a bit of a "hack", it might be sufficient for your implementation as it will be much easier for an end user to use.

Alerts that should appear adjacent to a person's name or address. If your organization doesn't use multiple character sets, you might want to consider using this functionality to implement critical person or premise alerts. For example, if you have a customer who's supported by a specific account representative, you could enter the account rep's name as the person's alternate name. If you do this, the account rep's name would appear in parenthesis following the customer's name. In addition, you can search for the customers supported by the account rep on Control Central by entering the account rep's name. This is a bit of a "hack", but it might prove useful for a variety of functions.

If your organization uses alternate representations of person name or address, set this flag to one of the following values:

- Use a value of Address if you only use alternate representations for premise addresses.
- Use a value of Name if you only use alternate representations for a person's primary names.
- Use a value of Name & Address if you use alternate representations for both premise addresses and person names.

The following points describe the ramifications of this flag in the system:

- If you support alternate representations of a person's primary name,
- The name grid on [Person - Main](#) allows you to specify an Alternate name for the person.
- If you use the base package [name formatting algorithm](#), a person's name will be shown throughout most of the system in the format AAA (BBB), where AAA is the person's primary name and BBB is the person's alternate name. Note, this

format does not apply to names that appear in search results (i.e., the alternate name is not concatenated to the main name in search results; however you can search for information using the alternate name).

- Most of the system's person name-oriented searches will allow users to use both a person's primary and alternate names to search for information.
- If you support alternate representations of a premise's address,
- A new tab is available on the [Premise](#) page that allows a user to define an alternate address for a premise.
- If you use the base package [address formatting algorithm](#), a premise's address will be shown throughout most of the system in the format AAA (BBB), where AAA is the premise's primary address and BBB is the premise's alternate address.
- Most of the system's premise-oriented searches will allow users to use both a premise's primary and alternate addresses to search for information.

If the CTI Integration flag has been enabled, set the **CTI Integration** flag to Yes if your organization integrates with an external computer telephony integration (CTI) system that supports a "get next caller in the queue" function. If this flag is set to Yes, then the **Next Call** button will appear in the application toolbar allowing customer service representatives to request the next customer waiting in the queue to speak to a CSR.

WARNING:

In order to improve response times, installation options are cached the first time they are used after a web server is started. If you change this field's option and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. Refer to [Caching Overview](#) for information on how to clear the system login cache (this is the cache in which installation options are stored).

Installation Options - Person

Select **Admin > General > Installation Options** and use the **Person** page to define person-specific installation options.

Description of Page

Use the **Person ID Usage** to indicate whether or not at least one form of identification is Required or Optional when a new person is added.

Each form of identification has an identifier type. For persons that are humans (as defined by the person type), the system defaults the identifier type defined in **Identifier Type (Person)**. For persons that are businesses (as defined by the person type), the system defaults the identifier type defined in **Identifier Type (Business)**.

Installation Options - Account

Select **Admin > General > Installation Options** and use the **Account** page to define account-specific installation options.

Description of Page

When a new account is added, the system requires it have a customer class. If the main customer linked to the account is a human (as defined by the customer's person type), the system defaults the customer class defined in **Customer Class (Person)**. For persons that are businesses (as defined by the person type), the system defaults the customer class defined in **Customer Class (Business)**. For more information, refer to [Setting Up Customer Classes](#).

In addition to requiring a customer class when a new customer is added, the system also requires a "main customer" (i.e., a reference to a person who is identified as the main customer for the account). Enter the default **Account Relationship Type Code** to be used to define the main customer relationship. For more information, refer to [Setting Up Account Relationship Codes](#).

Enter the default **Bill Route Type** to be used to define how bills should be routed to a customer. For more information, refer to [Setting Up Bill Route Types](#).

Enter the default **Quote Route Type** to be used to define how quotes should be routed to a customer. For more information, refer to [Setting Up Quote Route Types](#).

If the number of pending start and pending stop service agreements exceeds the **Start Stop Detail Threshold** for an account, it is considered a large account for start stop purposes. Refer to [Start/Stop Maintenance](#) for more information.

Installation Options - Billing

Select **Admin > General > Installation Options** and use the **Billing** page to define billing-specific installation options.

Description of Page

The **Bill Segment Freeze Option** controls when a service agreement's balance and the general ledger is affected by bill segments and certain types of adjustments. Refer to [Preventing SA Balances And The GL From Being Impacted Until Bill Completion](#) to understand the significance of this option.

The **Accounting Date Freeze Option** controls how the accounting date defined on financial transactions is populated. Refer to [Forcing The Freeze Date To Be Used As The Accounting Date](#) to understand the significance of this option.

Define the **Rollover Threshold Factor** used by billing to determine if a register's consumption is sensible. This value is used as follows:

- Whenever billing calculates a meter's register's consumption, it compares it to a value equal to X times the register's maximum capacity (where X is the Rollover Threshold Factor).
- If consumption exceeds this value, a bill segment error is generated. If this consumptive value is correct, a user will need to override the consumption value billed on the bill segment (billing will never use such a read).

Define the **Minimum Amount for Final Bill**. If a final bill is less than this amount, the bill is still produced; it's just not printed.

Typically, the system sets a bill's Bill Date equal to the date on which it is completed. If you want to be able to specify a bill's Bill Date when you complete a bill, turn on **User Can Override Bill Date**. You would only want to override the bill date if you are setting up sample bills from historical period whose bill date needs to reflect the respective historical period.

Turn on **Use High Low Failures on Bill** if the system should mark meter reads that fail high / low checks as billable. Turn off this switch if such reads should not be used by billing. Users may override this default value on a specific read. Refer to [Review High / Low](#) for more information.

Base Time is used by interval billing algorithms to determine the effective start and end times for a given period. The **Start Day Option** further defines how to use the base time, indicating whether the base time is for the Current Day or for the Previous Day. Refer to [Start and End Times for Billing](#) for more information.

Turn on **Use Alternative Bill ID** if your implementation uses assigned document numbers or sequential bill numbers. In the **Alternative Bill ID Option** list:

- Select **Document Numbers** if you require a system-assigned document number for each bill in addition to the Bill Id, which is a system-assigned random number used as the bill's primary identifier. Refer to [Document Numbers](#) for more information.
- Select **Sequential Bill Numbers** if you require a system-assigned unique sequential number for each bill in addition to the Bill Id, which is a system-assigned random number used as the bill's primary identifier. Refer to [Sequential Bill Numbers](#) for more information

NOTE:

Document Number Algorithms. In addition to turning on **Use Alternative Bill ID** and specifying the **Alternative Bill ID Option**, the [Document Number](#) and [Document Number Details](#) algorithms must be enabled on the [Installation](#) record. These algorithms contain the logic used by the system to assign a document number to a bill.

The **Bill Correction** option lets you control whether your implementation uses Credit Notes or Correction Notes. Select the **Credit Note** option if you require bill segment cancellation details to be presented to the customer on a separate bill (referred to as a credit note). Refer to [Credit Notes](#) for more information. Select the **Correction Note** option if you require bill segment cancellation details and bill segment rebill details to be presented to the customer on a separate bill (referred to as a correction note). Refer to [Correction Notes](#) for more information.

NOTE:

Credit Notes or Correction Notes. The Bill Correction option on the Installation table controls whether Credit Notes or Correction Notes are allowed. If your implementation uses Correction Notes, the override label on the following should be customized accordingly:

Lookup value CRNT on the customizable lookup field TXN_FLTR_TYPE_FLG (this lookup value is used on the Match Event Page and Account Bill History transactions)

Lookup value CR on the customizable lookup field PYCAN_SYS_DFLT_FLG (this lookup value is used on the Pay Cancel Reason transaction)

Metadata field CR_NOTE_FR_BILL_ID (this field is used on the Bill Search Page)

The **Autopay Creation Option** controls when automatic payments are created, distributed, and frozen. This option allows you to control when automatic payments will affect customer's balances and when their financial impact affects the general ledger. Refer to [How And When Are Automatic Payments Created](#) to understand the significance of this option.

Installation Options - C&C

Select **Admin > General > Installation Options** and use the **C&C** page to define credit and collections-specific installation options.

Description of Page

When you look at an account or service agreement's debt, the system shows the respective age of each piece of outstanding debt. The **Oldest Bucket Age (Days)** defines the debt age after which the system groups all outstanding debt together. For example, if this field is 180:

- The exact age of each element of debt that is less than 180 days old would be shown as a separate line item in the aged debt information.
- All debt older than 180 days would be amalgamated into a single "bucket".

Oldest Bucket Age (Days) also has another use - it defines the age of financial transactions that are considered by the background process that marks old debt as "redundant". This batch process is referred to by the batch code of REDSAAMT.

WARNING:

If you change the value of **Oldest Bucket Age (Days)** after debt has been marked as "redundant" by REDSAAMT, the system will NOT re-age the old debt (i.e., once a financial transaction has been marked as "redundant", it is "redundant" forever).

Enter what you consider to be an excellent credit rating in **Beginning Credit Rating**. Collection events can cause an account's credit rating to decrease. When an account's credit rating falls below a certain level, different collection processes may ensue.

Use **Beginning Cash-Only Score** to define the cash-only score for accounts with a perfect payment history (i.e., one without non-sufficient funds). When you cancel a payment tender and use a cancellation reason marked as NSF, the system will cause the account's cash-only score to increase by the value on the payment cancellation reason.

Use **Credit Rating Threshold** to define when an account's credit rating becomes risky. When an account's credit rating falls beneath the Credit Rating Threshold, the system will:

- Assuming you've enabled the Control Central alert algorithm, [C1-CRRT-ACCT](#), an alert displays when an account's credit rating falls below the credit rating threshold on the CIS installation table. This algorithm is plugged-in on the [installation record](#).
- Subject the account's debt to different collection criteria. For more information, refer to [Designing Your Collection Class Control Overrides](#).

Use **Cash-Only Threshold** to define the number of cash-only points a customer must have before the system warns the CSR accepting payments that the account is cash-only.

Installation Options - Financial Transaction

Select **Admin > General > Installation Options** and use the **Financial Transaction** page to define financial transaction installation options.

Description of Page

Use **G/L Batch Code** to define the batch process that is used to interface your financial transactions to your general ledger. The process is snapped on FT download records by the GLS background process.

Use **A/P Batch Code** to define the batch process that is used to interface your check requests (initiated with adjustments with an adjustment type that reference an A/P request type) to you accounts payable system.

Use **Fund Accounting** to indicate if [fund accounting](#) is Practiced or Not Practiced at your organization.

Use **Alternate Currency** to indicate if your organization accepts customer payments in currencies other than the account's currency.

FASTPATH:

Refer to [Alternate Currency Payments](#) to understand the significance of this option.

Installation Options - Algorithms

Select **Admin > General > Installation Options** and use the **Algorithms** page to define specific system events.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Account Information	Optional	We use the term "Account information" to describe the basic information that appears throughout the system to describe an account. The data that appears in "account information" is constructed using this algorithm. Plug an algorithm into this spot to override the system default "Account information". Click here to see the algorithm types available for this system event.

Adjustment Information	Optional	<p>We use the term "Adjustment information" to describe the basic information that appears throughout the system to describe an adjustment. The data that appears in "Adjustment information" is constructed using this algorithm.</p> <p>Plug an algorithm into this spot to override the system default "Adjustment information".</p> <p>Note: This algorithm may be further overridden by an "Adjustment information" plug-in on the Adjustment Type. Refer to Adjustment Type for how algorithms of this type are used.</p> <p>Click here to see the algorithm types available for this system event.</p>
Appointment Information	Required	<p>We use the term "Appointment information" to describe the basic information that appears throughout the system to describe an appointment. The data that appears in "appointment information" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Automatic Payment Creation	Required if you allow customers to pay automatically	<p>This algorithm is executed to create automatic payments whenever the system creates automatic payments. Refer to How And When Are Automatic Payments Created for the details.</p> <p>Click here to see the algorithm types available for this system event.</p>
Bill Information	Required	<p>We use the term "Bill information" to describe the basic information that appears throughout the system to describe a bill. The data that appears in "bill information" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Bill Segment Information	Optional	<p>We use the term "Bill segment information" to describe the basic information that appears throughout the system to describe a bill segment. The data that appears in "bill segment information" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Case Information	Optional	<p>We use the term "Case information" to describe the basic information that appears throughout the system to describe a case. The data that appears in "case information" is constructed using this algorithm.</p>

		<p>Plug an algorithm into this spot to override the system default "Case information".</p> <p>Note: This algorithm may be further overridden by a "Case information" plug-in on the Case Type. Refer to Case Type for how algorithms of this type are used.</p> <p>Click here to see the algorithm types available for this system event.</p>
Collection Agency Referral Information	Optional	<p>We use the term "Collection Agency Referral information" to describe the basic information that appears throughout the system to describe a collection agency referral.</p> <p>Plug an algorithm into this spot to override the system default "collection agency referral information".</p> <p>Click here to see the algorithm types available for this system event.</p>
Collection Process Additional Information	Optional	<p>This algorithm displays additional information related to a collection process in a special field on the collection process main page.</p> <p>Click here to see the algorithm types available for this system event.</p>
Control Central Alert	Optional	<p>There are two types of alerts that appear in the Alert Zone and on Payment Event - Main: 1) hard-coded system alerts and 2) alerts constructed by plug-in algorithms. You cannot change the hard-coded alerts (see the Alert Zone for the complete list). However, by plugging in this type of algorithm you can introduce additional alerts.</p> <p>An error displays if more than 60 alerts are generated for an account by plug-in algorithms.</p> <p>Click here to see the algorithm types available for this system event.</p>
Credit Rating "Created By" Information	Required	<p>The data that appears in the credit rating "created by" information is constructed using this algorithm.</p> <p>Refer to Account - C&C for more information about the credit rating.</p> <p>Click here to see the algorithm types available for this system event.</p>
Credit Rating History Information	Optional	<p>We use the term Credit Rating History information to describe the basic information that appears throughout the system to describe a credit rating history entry.</p> <p>Plug an algorithm into this spot to override the system default "credit rating history information".</p>

		Click here to see the algorithm types available for this system event.
Document Number	Optional	<p>If document numbers have been enabled on the installation record, this algorithm type assigns a document number to a bill or payment event.</p> <p>Click here to see the algorithm types available for this system event.</p>
Document Number Details	Optional	<p>If document numbers have been enabled on the installation record, this algorithm type is responsible for returning the details used to construct the document number.</p> <p>Click here to see the algorithm types available for this system event.</p>
Determine Open Item Bill Amounts	Required if you use overdue functionality to collect on bills	<p>This algorithm is responsible for determining the unpaid amount of an open-item bill. It can also be used to return the unpaid amount for a specific SA on a bill.</p> <p>Click here to see the algorithm types available for this system event.</p>
FA Additional Information	Optional	<p>This algorithm displays additional information related to a field activity in a special field called Additional Info on the field activity main page.</p> <p>For example, contact information linked to the field activity's field order may be displayed.</p> <p>Click here to see the algorithm types available for this system event.</p>
FA Information	Required	<p>We use the term FA information to describe the basic information that appears throughout the system to describe a field activity. The data that appears in "FA information" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Item Information	Required if you have items	<p>We use the term "Item info" to describe the basic information that appears throughout the system to describe an item. The data that appears in "Item info" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Meter Information	Required if you have meters	<p>We use the term "Meter info" to describe the basic information that appears throughout the system to describe a meter. The data that appears in "Meter info" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>

Meter Read High Low Limits	Optional	<p>This algorithm is executed to calculate high and low limits for high / low check when a meter read is added to the system (whether through a batch upload or online).</p> <p>Click here to see the algorithm types available for this system event.</p>
Online Bill Display	Optional	<p>This algorithm constructs a PDF that contains the image of a bill. This algorithm is executed when the Display Bill button is clicked on the Bill page. Refer to Technical Implementation of Online Bill Image for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Online Field Order Image	Optional	<p>This algorithm constructs a PDF that contains the image of a field order. This algorithm is executed when the Display Field Order button is pressed on the Field Order page. Refer to Technical Implementation of Online Field Order Image for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Online Letter Image	Optional	<p>This algorithm constructs a PDF that contains the image of a letter. This algorithm is executed when the Display Letter button is pressed on Customer Contact - Main. Refer to Technical Implementation of Online Letter Image for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Online Quote Image	Optional	<p>This algorithm constructs a PDF that contains the image of a quote. This algorithm is executed when the Display Quote button is pressed on Quote - Main. Refer to Technical Implementation of Online Quote Image for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Online Statement Image	Optional	<p>This algorithm constructs a PDF that contains the image of a statement. This algorithm is executed when the Display Statement button is pressed on Statement - Main. Refer to Technical Implementation of Online Statement Image for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Override Proration Factors	Optional	<p>This algorithm is only used if your organization has unusual rate proration requirements that necessitate the overriding of the base package proration logic. For example, you may have certain calculation</p>

rules whose charges should never be prorated. Refer to [Overriding Proration Factors](#) for more information.

Click [here](#) to see the algorithm types available for this system event.

Override Seasonal Proration	Optional	<p>This algorithm is only used if your organization has unusual method of determining the seasons for your calculation rules. For example, you may determine the seasonal boundaries for a calculation rule based on the scheduled meter read date associated with the bill cycle.</p> <p>Click here to see the algorithm types available for this system event.</p>
Payment Amount Calculation	Required	<p>This algorithm is executed to calculate the amount of an automatic payment for a bill for an account with an active auto pay option. Refer to How And When Are Automatic Payments Created for more information on automatic payments. This algorithm is also executed to default the amount of a manually added payment. Refer to How To Add A New Payment Event for more information on adding a payment manually.</p> <p>Click here to see the algorithm types available for this system event.</p>
Payment Information	Required	<p>We use the term "payment information" to describe the basic information that appears throughout the system to describe a payment. The data that appears in "payment information" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Person Information	Required	<p>In most parts of the system, a person's Main name is displayed to describe a person. However, several transactions do not use this method. Rather, these transactions call the algorithm that's plugged into this spot to construct the person's name. Refer to the description of the Alternate Representation flag on the Main tab for a list of these transactions and for the rationale behind this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Person Name Validation	Required	<p>The format of names entered on Person - Main and Order - Main is validated using this algorithm.</p>

		Click here to see the algorithm types available for this system event.
Premise Information	Required	<p>We use the term "premise info" to describe the basic information that appears throughout the system to describe a premise. The data that appears in "premise info" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
Reporting Tool	Optional	<p>If your installation has integrated with a third party reporting tool, you may wish to allow your users to submit reports on-line using report submission or to review report history on-line. This algorithm is used by the two on-line reporting pages to properly invoke the reporting tool from within the system.</p> <p>Click here to see the algorithm types available for this system event.</p>
SA Information	Optional	<p>We use the term "SA information" to describe the basic information that appears throughout the system to describe a service agreement. The data that appears in "SA information" is constructed using this algorithm.</p> <p>Plug an algorithm into this spot to override the system default "SA information".</p> <p>Note: This algorithm may be further overridden by an "SA information" plug-in on the SA Type. Refer to SA Type - Algorithms for how algorithms of this type are used.</p> <p>Click here to see the algorithm types available for this system event.</p>
Severance Process Cancellation	Optional	<p>This algorithm is executed to perform additional processing when the system cancels a severance process.</p> <p>Note: This algorithm is executed before the Severance Process Template - Post Cancel Algorithm is executed. Canceling a severance process on-line manually does not execute this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>
SP Information	Required	<p>We use the term "SP info" to describe the basic information that appears throughout the system to describe a service point. The data that appears in "SP info" is constructed using this algorithm.</p> <p>Click here to see the algorithm types available for this system event.</p>

Defining Customer Languages

As described under [Defining Languages](#), you define the language in which each user see the system. In addition to defining each user's language, the system allows you to define each customer's preferred language. For example, one customer can receive bills in English whereas another customer could receive their bills in Chinese.

Each customer's language is defined by the [language code](#) on their [person record](#). Bills, adjustments and other system-generated records will then be done in the language of the main customer of the account. In addition, the language code is passed on to all customer-facing interfaces, such as letter requests and bill print.

NOTE:

You can define Rates in multiple languages - when a bill is generated, the line-item descriptions are generated and stored in the account's main customer's language of choice. Anyone who subsequently views these bills can only see the descriptions in that language.

To support bills and other correspondence, you must also provide translations of standard bill stock and letters. This must be handled by your printing software vendor.

Defining Accounting Calendars

Accounting calendars determine the accounting period to which a financial transaction will be booked. The following points describe how the system determines a financial transaction's account period:

- Every financial transaction references an accounting date and its service agreement
- Every service agreement references a service agreement type
- Every service agreement type references a GL division
- Every GL division references an accounting calendar
- The accounting calendar contains the cross-reference between the accounting date specified on the financial transaction and related accounting period in your general ledger

WARNING:

This information must be the same as the information in your financial database.

To add or review an accounting calendar, choose **Admin > Accounting Calendar > Search**.

Description of Page

Enter a unique **Calendar ID** and **Description** for the calendar.

Enter the **Number Of Periods** for the calendar. Don't count the adjustment period, if you use one, or any special "system" periods.

Specify the **Fiscal Year**, each **AccountingPeriod** in that year, a **Period Description**, the **Begin Date** and the **End Date**.

When you enter begin and end dates, you can define monthly calendar periods or any fiscal period that matches your accounting calendar (weekly, bimonthly) as long as the begin and end dates of successive periods do not overlap. Every day of the year must be included in a period; do not leave gaps between period dates.

For each fiscal period, enter the **Open From Date** and **Open To Date**. These dates define when that particular business dates are open for posting financial transactions to that fiscal period. For example, you might calculate a bill on Sept 1 for usage recorded on 31 August. To post this financial transaction in the August period, you must keep it open through Sept 1.

As time passes, you will need to return to this transaction to manually enter ensuing years. You can enter several years at a time or incorporate the task into end-of-year system maintenance.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CAL_GL](#)

Defining General Ledger Divisions

There are two types of Divisions referenced in the system: a CIS Division and a GL Division. This is a rather powerful structure, but it can be confusing.

- General Ledger divisions typically comprise individual entities (e.g., companies) in your general ledger. You must set up a GL division for each such entity. The GL division's sole purpose in the system is to define the accounting period associated with financial transactions linked to service agreements associated with the GL division (service agreements are associated with GL divisions via their SA type). The system cares about accounting periods in order to prevent a user from booking moneys to closed periods. It also uses accounting periods when it produces the flat file that contains the consolidated journal entry that is interfaced to your general ledger (refer to [The GL Interface](#) for more information).

NOTE:

When determining how many GL Divisions you need, be sure to consider your general ledger and how your chart-of-accounts is structured. You will typically have one GL division for each "company" in your general ledger.

-
- A CIS division is typically associated with a jurisdiction. The definition of a jurisdiction is a geographic-oriented entity with unique business rules. For example, if you conduct business in California and Nevada, and each state has different collection rules, you will need a separate jurisdiction for each state. You must set up a CIS division for each jurisdiction in which you conduct business.

FASTPATH:

Refer to [Setting Up CIS Divisions](#) for information about CIS Divisions.

To define a general ledger division, select **Admin > General Ledger Division**.

Description of Page

Enter a unique **GL Division** for the general ledger division.

Enter a **Description** of this general ledger division.

Define the accounting **Calendar ID** that controls how to convert an FT's accounting date into an accounting period. Refer to [Defining Accounting Calendars](#) for more information.

You may define a **Currency Code** for the GL division. Note that the system does not use this currency code.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_GL_DIVISION](#).

Defining Banks & Bank Accounts

The topics in this section describe how to maintain your implementation's bank accounts.

Bank - Main

To review Banks choose **Admin > Bank > Search**.

Description of Page

Enter a unique **Bank Code** and **Description** for the bank.

The **Bank Accounts** collection displays the bank accounts currently linked to this bank code. Use the drill down button to view more details or to modify the bank account details. Alternatively, you may navigate to the Bank Account tab and scroll to the desired bank account.

Bank - Bank Account

To review Bank Accounts for a Bank, choose **Admin > Financial > Bank > Search** and then navigate to the **Bank Account** page.

Description of Page

Use the **Bank Accounts** tab to define the attributes of each bank account. For each account, enter the following information:

- Enter a **Bank AccountKey** to identify an Account at a Bank. You may have more than one account at a given bank, and you may have accounts at more than one bank. This code will allow the system to easily identify a specific account.
- Enter a **Description** to appear on prompt lists, inquiries, and reports.
- Enter the **Account Number**, **Check Digit** and if needed, the **Branch ID** of the bank where the account is held.
- Enter the **CurrencyCode** for the currency in which the account is denominated.
- Use **DFI ID** to define the Depository Financial Institution ID that is interfaced to the automatic payment-processing agent as part of the automatic payment interface.
- Enter the **Distribution Code** to be used for cash GL distributions when a payment is frozen or canceled.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BANK_ACCOUNT](#).

Defining Issuing Centers

This section provides information about defining issuing centers that are used to assign document numbers to bills and payment events. An issuing center should be configured for each location that issues bills. The installation record [Document Number](#) and [Document Number Details](#) algorithms contain the logic used by the system to assign a document number to a bill. To set up an issuing center, open **Admin > General > Issuing Center > Add**.

FASTPATH:

Refer to [Document Numbers](#) for information about document number assignment.

NOTE:

This section is only relevant for some organizations. The information in this section is only relevant if your organization indicated on the installation record that it uses document numbers as an alternative bill id. If your organization does not use document numbers as an alternative bill id, then no other setup is required.

The topics in this section describe the base-package zones that appear on the Issuing Center portal.

Actions

This is a standard [actions zone](#).

If the issuing center is in a state that has valid next states, buttons to transition to each appropriate next state are displayed.

Issuing Center List

The Issuing Center [List zone](#) lists every issuing center. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent issuing center.
- Click the Add link in the zone's title bar to add a new issuing center.

Issuing Center

The Issuing Center zone contains display-only information about an issuing center, including its current and historic branches. This zone appears when an issuing center has been broadcast from Issuing Center List zone or if this portal is opened via a drill down from another page.

Please see the zone's help text for information about this zone's fields.

Issuing Center Log

This is a standard [log zone](#).

Setting Up Service Types

You will have one service type for each type of service you provide to your customers. If we assume that your organization sells electricity, gas and water, you will need three service types for these services. In addition, you will probably want a catch all service type of Other to put on SA types used for write-offs, payment arrangements and deposits.

NOTE:

Non Service Point-Oriented Service Types. You may require additional service types if you have non service point-oriented services, e.g., land leases and deposits. Refer to [Service Segmentation](#) for more information.

This page is also used to define valid facility levels for a service type. You may wonder, What is a facility level? Every type of service tends to use a different mapping philosophy to designate the facility hierarchy that supplies service to the service point. For example, electric service typically uses a substation / feeder / node facility hierarchy to define how electricity is supplied to a service point (the substation is the highest level in the hierarchy, the feeder comes next, and finally the node). On the other hand, gas service uses a city gate / main / feeder hierarchy.

If your organization maintains this type of information on service points, you will set up your facilities and their interrelationships. On this page you set up the number and type of facility levels used for every service and you define the valid values for each facility level. On the [Facility Level 1 & 2](#) and [Facility Level 2 & 3](#) pages, you define the values that may coexist in each level. After these set up tasks are complete, you're ready to enter facility levels on your service points.

NOTE:

A service point's facility levels are used to help pinpoint problems and dispatch service crews during outages.

The topics in this section describe how to set up service types and facility levels.

Service Type - Main

To define service types and the types of facility levels, select **Admin > General > Service Type > Add**.

Description of Page

Enter a unique **Service Type** and **Description** for each service type.

Use the **Facility Level Names** collection to define the **Facility Level** and **Description** for each level in the service type's hierarchy. The description is used as the label prefixing the respective facility level on the Service Point Maintenance page.

Move to the **Level 1** tab to maintain the valid values for the highest facility level.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_SVC_TYPE](#).

Service Type - Level 1

Open **Admin > General > Service Type > Add** and navigate to the **Level 1** page to define the various facilities classified at the highest level.

Description of Page

You can optionally start the grid at a given **Facility Level 1**.

Enter a **Facility Level 1** code and a **Description** for each facility in the highest level.

Service Type - Level 2

Open **Admin > General > Service Type > Add** and navigate to the **Level 2** page to define the various facilities classified at the second level.

Description of Page

You can optionally start the grid at a given **Facility Level 2**.

Enter a **Facility Level 2** code and a **Description** for each facility at the second level.

Service Type - Level 3

Open **Admin > General > Service Type > Add** and navigate to the **Level 3** page to define the various facilities classified at the third level.

Description of Page

You can optionally start the grid at a given **Facility Level 3**.

Enter a **Facility Level 3** code and a **Description** for each facility at the third level.

Defining Service Tasks Options

This section describes concepts and procedures related to service tasks.

About Service Tasks

Service tasks are records that can be used to perform a variety of tasks. Examples include:

- Task-related records performed by users of other Oracle Utilities applications, such as Oracle Utilities Customer Self Service.
- Task-related records performed by Oracle Utilities Customer Care and Billing to manage specific processing, such as net energy metering true-ups, or prepaid billing to create bill segments for smart meter prepaid service agreements.

The main attribute used to define a service task is service task type. The service task type defines properties common to specific types of service tasks.

For more information, see [Searching and Viewing Service Tasks](#).

About Service Task Types

Service task types define properties common to specific types of service tasks. Service task types represent different types of tasks that can be performed by:

- Users of other Oracle Utilities applications, such as Oracle Utilities Customer Self Service. An example of a service task includes self service meter reads, in which users enter their own meter reads via the Customer Self Service application.
- Oracle Utilities Customer Care and Billing to manage specific processing. Examples include net energy metering true-ups, or prepaid billing to create bill segments for smart meter prepaid service agreement.

Service task types can be defined by the following attributes:

- **Service Task Type:** the name of the task type.
- **Service Task Type Business Object:** the business object that defines the behavior of the service task type.
- **Service Task Business Object:** the business object instantiated when service tasks of this type are created.
- **Service Task Class:** the category used to define service task types for reporting purposes (self-service, miscellaneous, etc).
- Other data based on the specific type of service task (such as minimum days in true-up period, billing frequency, or default payment method.)

For more information about defining service task types, see [Defining Service Task Types](#).

For more information about the service task types delivered with the application, see [Base Package Service Task Types](#).

Defining Service Task Types

To maintain existing service task event types, select **Admin > Service Task Type > Search**, then use standard actions to edit, duplicate, or delete a service task type.

To define a new service task type, follow these steps:

1. Select **Admin > Service Task Type > Add**.
2. Enter a name and a meaningful description for the service task type.

3. If needed, select the business object to use when creating service tasks of this type.
4. Select the service task class applicable to service tasks of this type, if applicable.
Some service tasks types used solely by Oracle Utilities Customer Care and Billing have a default class of **Miscellaneous**.
5. Enter a detailed description for the service task type.
6. Complete the remaining fields and sections, as needed.
7. If applicable, select a To Do type and corresponding To Do role to use when creating To Do entries related to service tasks of this type.
8. Click **Save**.

This service task type can now be used when service tasks are received from other Oracle Utilities applications, such as Oracle Utilities Customer Self Service, or service tasks created specifically by Oracle Utilities Customer Care and Billing. For more information about service task types, see [About Service Task Types](#).

For more information about the service task types delivered with the application, see [Base Package Service Task Types](#).

Base Package Service Task Types

This section provides descriptions of the service task types provided with the base package.

Service Task Type	Business Object	Detailed Description	Related Transaction Business Object
Appointment Notification Task Type	C1-NotifyAppointmentTaskType	This business object is used to capture the information to use in appointment notification processing.	Appointment Notification Task
Create Customer Contact Task Type	C1-CreateCustContactTaskType	This business object is used to capture the information to use in create customer contact task processing.	Customer Contact Request Task
Direct Debit Task Type	C1-DirectDebitMandateTaskType	This business object defines configuration information that is used for processing Single Euro Payments Area (SEPA) direct debit transactions.	Direct Debit Mandate Task
FA Completion Task Type	C1-FACompletionTaskType	This business object captures the attributes used in field activity completion task processing.	FA Completion Task
Field Activity Remark Task Type	C1-FieldActivityRemarkTaskType	This business object is used to capture the information to use in create field activity remark task processing.	Field Activity Remark Task
Form Task Type	C1-FormTaskType	This business object captures attributes relevant for a particular form type to be used when a self-service customer creates a form.	Form Task
Missed Appointment Notification Task Type	C1-NotifyMissedApptTaskType	This business object is used to capture the information to use in missed appointment processing.	Missed Appointment Notification Task

Notification Task Type	C1-NotifyTaskType	This business object captures information about a notification that is made available for sign-up by self-service users.	Notification Preferences
Net Energy Metering Task Type	C1-NEMTrueUpTaskType	This business object is used to maintain the various configuration options that are used by the true up monitor (TUM) business object's algorithms. It defines the length of the true up period as well as the adjustments types used during the true up process. For true up reversals, the adjustment cancel reason to use is also captured here.	True Up Monitor Task
Outage Call Type	C1-OutageCallType	This business object defines the behavior of an outage call type used to support trouble calls captured in CCB and integrated to NMS.	Outage Call
Payment Arrangement Task Type	C1-PATaskType	This business object defines the expected behavior for when a self-service user requests a payment arrangement.	Payment Arrangement Task
PPB Payment Notification Task Type	C1-PPBPaymentNotifyTaskType	This business object captures information about the Prepaid Billing Payment Request notification that is made available for sign-up by self-service users.	Notification Preferences
Prepay Biller Task Type	C1-PrepayBillerTaskType	This business object defines the behavior of a prepay biller task type used in smart meter prepay billing.	Prepay Biller Task
Product Offer Publish	C1-ProductOffer	This business object contains the product offer elements that are relevant when publishing product offers to Siebel Energy. It is used to read product offer information when building the initial and final snapshots for the sync request.	Not defined
Product Offer Task Type	C1-ProductOfferTaskType	This business object is used to define the behavior of a product offer. Product offers are configured in CCB and published to Siebel Energy.	Not defined
SA Creation Rule Publish	C1-SACreationRule	This business object contains the SA creation rule elements that are relevant when publishing product offers to Siebel Energy. It is used to read SA creation rule information when building the	Not defined

		initial and final snapshots for the sync request.	
SA Creation Rule Task Type	C1-SACreationRuleTaskType	This business object defines the behavior of the SA creation rules for a product offer. Each SA creation rule indicates the type of service agreement to create for the product offer.	Not defined
Service Request Task Type	C1-ServiceRequestTaskType	This business object is used as a parent service task type business object for service request related task types such as appointment notification, missed appointment and customer contact creation requests.	Not defined
Siebel Customer Maintenance Task Type	C1-CustomerMaintenanceTaskType	This business object is used in the Siebel integration. When a customer maintenance request is received from Siebel, an inbound service determines the service task type to use for the event.	Siebel Customer Maintenance Task
Start/Stop Task Type	C1-StartStopTaskType	This business object defines the expected behavior for when a self-service user requests to start, stop or transfer service.	Start/Stop Request Task

Defining Financial Transaction Options

Bills, payments and adjustments share one very important trait - they affect how much your customers owe. This section explains the financial design of the system and describes how to set up the tables that control the financial impact of these transactions.

NOTE:

The tables in this section are the first of many that must be set up before you can create bills and apply payments. In this section, we limit the discussion to those tables that control the financial impact of bills, payments and adjustments. In later sections, we describe the tables that control other billing-related functions like meter reading and rates. It is only after all of these tables are set up that you will be able to generate the various financial transactions.

The Financial Big Picture

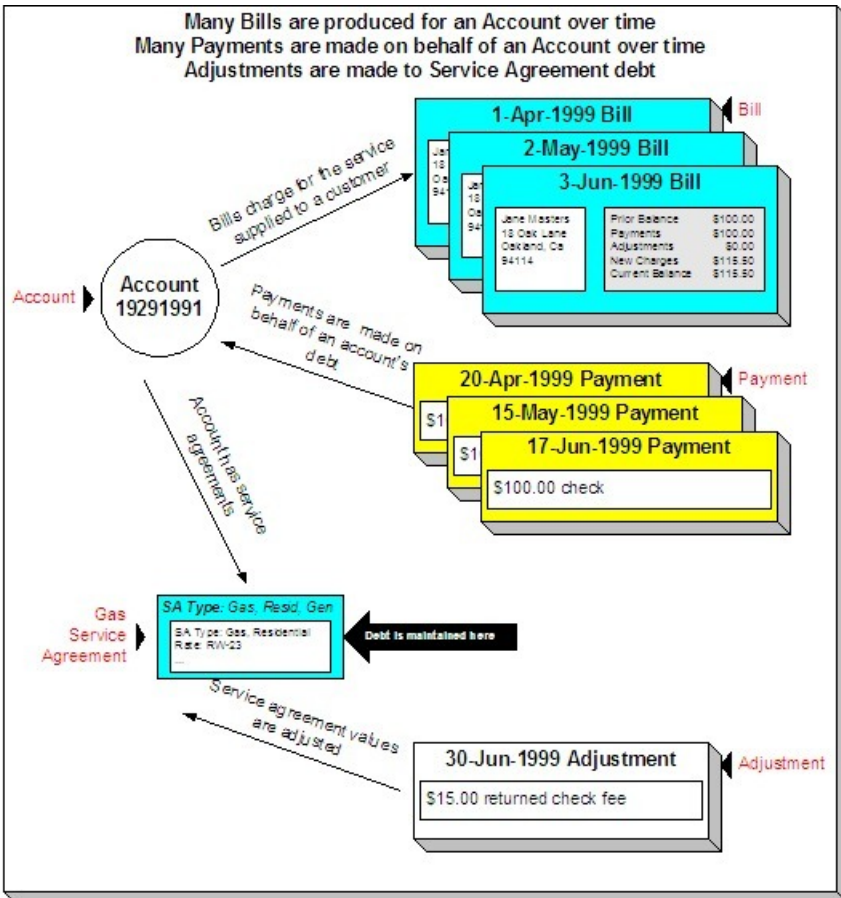
This section provides an overview of the relationship between an account and the various financial transactions that influence how much a customer owes.

WARNING:

If your organization practices cash accounting for payables (i.e., you only pay the taxing authority when you get paid), refer to [Payables Cash Accounting](#). If your organization practices open-item accounting (i.e., payments must be matched to bills), refer to [Open Item Accounting](#).

Bills, Payments & Adjustments

The following diagram illustrates the relationship between an account and its financial transactions:



The following concepts are illustrated above:

Bills are produced for accounts Over time, many bills may be produced for an account. For more information about a bill, see [Bill Details](#).

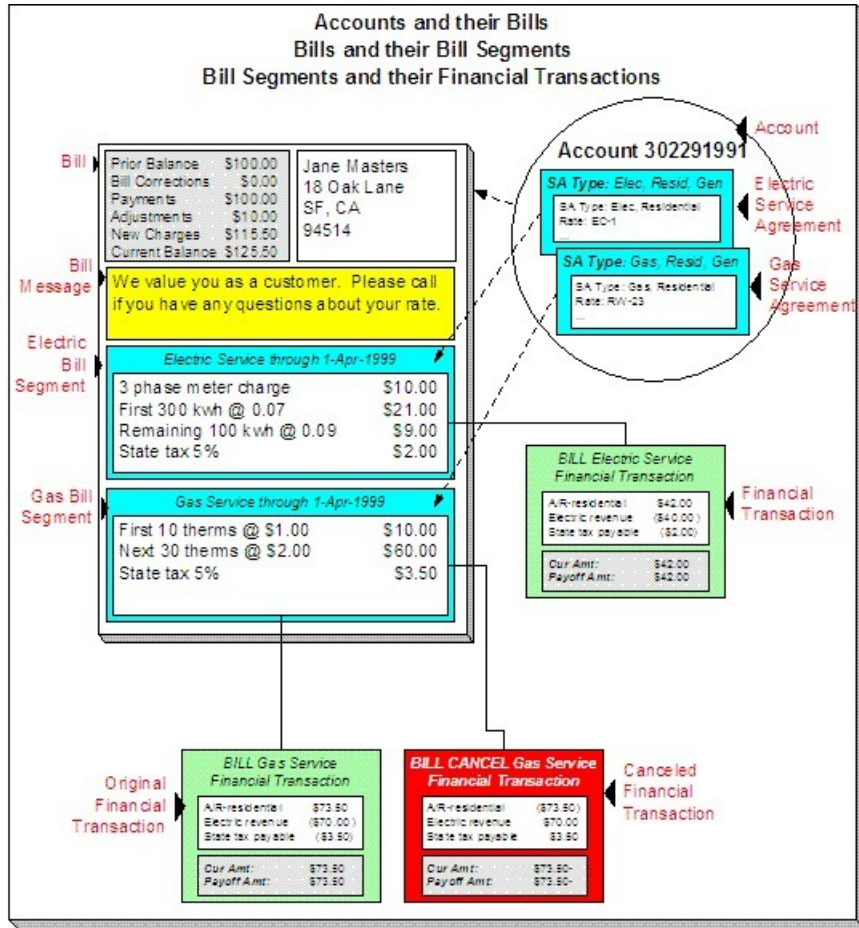
Payments are made for accounts Over time, many payments may be applied to an account's debt. For more information about a payment, see [Payment Details](#).

Service agreements have debt The system maintains debt on each individual service agreement. An account's debt is the sum of its service agreements' debt.

Service agreements are adjusted Over time, the debt that is stored on an account's service agreement(s) may be adjusted. For more information about an adjustment, see [Adjustment Details](#).

Bill Details

The following diagram illustrates the relationship between an account and its bills:



The following concepts are illustrated above:

A bill is produced for an account Over time, many bills are produced for an account. Bills charge for the services supplied to a customer. The above illustration shows a single bill.

Bills contain bill segments A bill typically contains one bill segment for every active service agreement linked to its account. The only time this is not true is when service agreements for different frequencies exist. For example, an account with a monthly and a quarterly service agreement will only have 4 bills a year that contain both bill segments; the other months' bills will contain a single bill segment for the monthly service agreement.

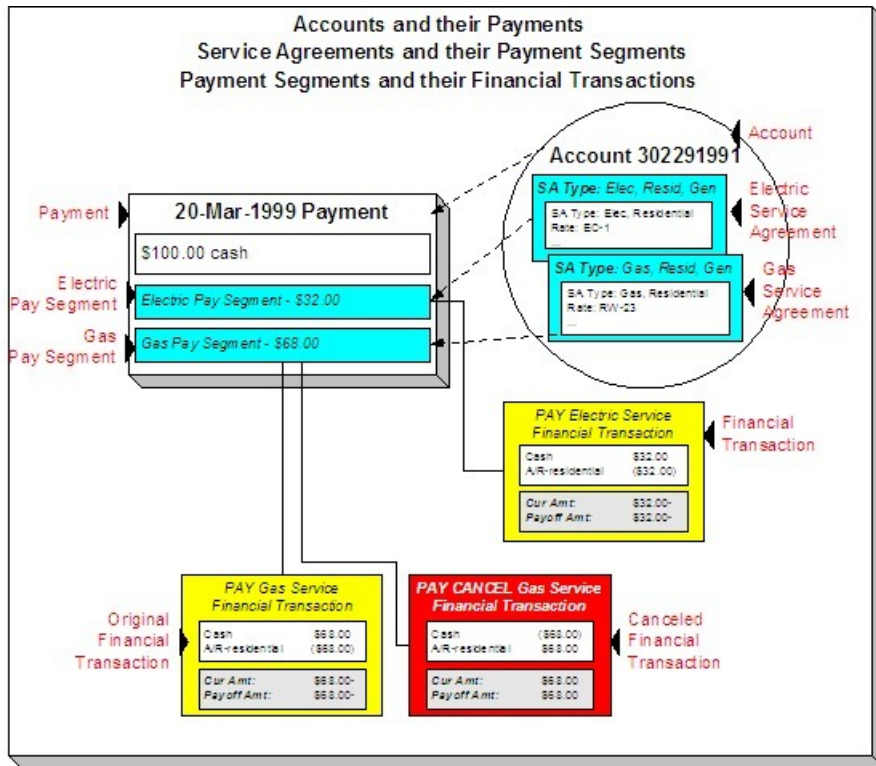
Bill segments contain calculation details A bill segment contains information showing how the segment was calculated and how it should be printed on the customer's bill.

A bill segment has a financial transaction A bill segment has a related financial transaction. A financial transaction contains the financial effects of the bill segment on the service agreement's current and payoff balances and on the general ledger.

Canceling a bill cancels the financial tran. If the bill segment is eventually cancelled, another financial transaction will be linked to the bill segment to reverse its original financial transaction. The cancellation financial transaction appears on the next bill produced for the account as a bill correction.

Payment Details

The following diagram illustrates the relationship between an account and its payments:



The following concepts are illustrated above:

Payments are made for accounts Over time, many payments may be applied to an account's debt. The above illustration shows a single payment.

Payments contain payment segments A payment contains one payment segment for every service agreement to which the payment is distributed. For a customer who pays in full, the number of payment segments will coincide with the number of bill segments on the bill being paid.

A pay. segment has a financial transaction A payment segment has a related financial transaction. A financial transaction contains the financial effects of the segment on the service agreement's current and payoff balances and on the general ledger.

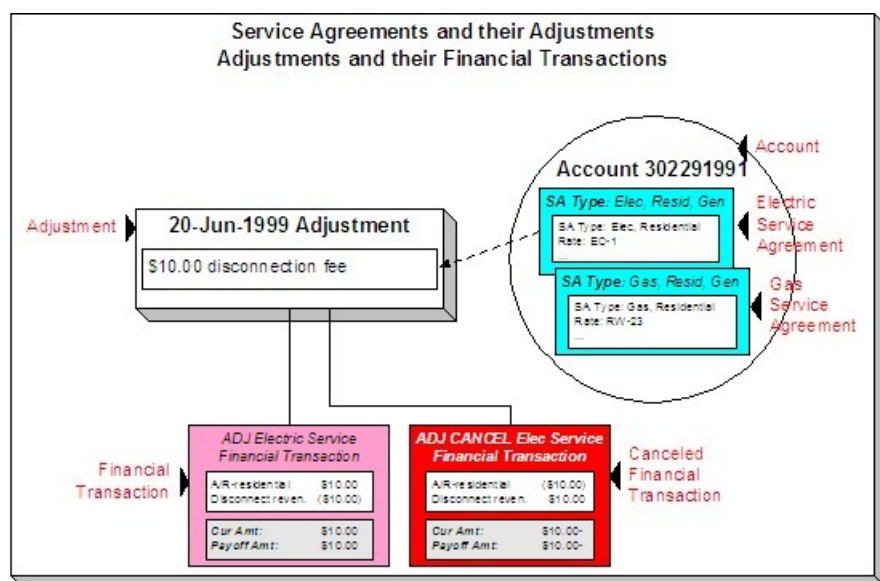
Canceling a payment cancels the fin. tran. If the payment is eventually cancelled, another financial transaction will be linked to the related payment segment(s) to reverse their financial effect. The cancellation financial transaction appears on the next bill produced for the account as a negative payment.

FASTPATH:

A payment cannot be applied to an account's debt without an associated payment event. Refer to [The Big Picture of Payments](#) for more information.

Adjustment Details

The following diagram illustrates the relationship between an account and its adjustments:



The following concepts are illustrated above:

Service agreements have adjustments Over time, a service agreement may have many adjustments. The above illustration shows a single adjustment to one of the account's service agreements.

An adjustment has a financial transaction An adjustment has a related financial transaction. The financial transaction contains the financial effects of the adjustment on the service agreement's debt and on the general ledger.

Canceling an adjust. cancels the fin. tran. If the adjustment is eventually canceled, another financial transaction will be linked to the adjustment to reverse its financial effect. The cancellation financial transaction appears on the next bill produced for the account as an adjustment.

Current Amount versus Payoff Amount

A financial transaction contains two very important attributes: payoff amount and current amount. These attributes contain the grand total of how much the customer owes.

- Current amount contains how much the customer THINKS THEY OWE.
- Payoff amount contains how much the customer REALLY OWES.

You may be wondering when these two values can be different? Well, for most financial transactions, these values are the same. These values differ under the following situations:

- When a bill segment charges a customer for a charitable contribution, payoff amount will be zero because the customer doesn't really owe anything (they don't have to contribute if they don't want to). Current amount will be equal to the agreed charitable contribution amount (the customer thinks they owe the contribution).
- When a bill segment charges a customer for a deposit, payoff amount will be zero because the customer doesn't really owe anything (billed deposits are typically not viewed as being a receivable). Current amount will be equal to the amount billed (the customer thinks they owe the deposit amount).
- When a bill segment charges a customer who participates in a levelized payment program (e.g., budget billing or non-billed budgets) the two "amounts due" will contain different values. Payoff amount is equal to how much the customer really owes for the service they consumed; current amount is equal to how much they think they owe in accordance with their monthly budget.

A perhaps easier way to view these two attributes is to consider payoff amount as the "cash out amount", i.e., the amount the customer would owe the utility if they wanted to clear up all debt. The current amount contains the amount the customer

thinks they owe. If you're still struggling with the difference, think about your monthly Visa bill: it contains a monthly minimum payment and the total amount owed. The minimum payment is the current amount; the total amount owed is the payoff amount.

The topics in this section provide more information about these two fields.

What Controls What Gets Booked To Current And Payoff Amount?

As described in [Bill Details](#), every bill segment has a sibling financial transaction. The financial transaction defines the bill segment's affect on current and payoff amounts due. The system populates these two fields as per the Financial Transaction Algorithm defined on the bill segment's bill segment type.

FASTPATH:

For more information, refer to [Billing - Current Balance versus Payoff Balance](#) and [Designing and Defining Bill Segment Types](#).

As described in [Payment Details](#), every payment segment has a sibling financial transaction. The financial transaction defines the payment segment's affect on current and payoff amounts due. The system populates these two fields as per the Financial Transaction Algorithm defined on the payment segment's payment segment type.

FASTPATH:

For more information, refer to [Payment - Current Balance versus Payoff Balance](#) and [Setting Up Payment Segment Types](#).

As described in [Adjustment Details](#), every adjustment has a sibling financial transaction. The financial transaction defines the adjustment's affect on current and payoff amounts due. The system populates these two fields as per the Financial Transaction Algorithm defined on the adjustment's adjustment type.

FASTPATH:

For more information, refer to [Adjustments - Current Balance versus Payoff Balance](#) and [Setting Up Adjustment Types](#).

Arrears

The system keeps track of the age of each customer's debt to the day. For example, if a customer hasn't paid their last two bills, the customer's aged debt might look as follows:

- \$124.50: 22 days old
- \$213.41: 51 days old

Please be aware that it is the current balance (i.e., what the customer thinks they owe) that is aged. Also keep in mind that the moment an FT is frozen, it impacts a customer's current balance.

The system represents aged debt in a variety of ways of the various transactions in the system. On the [Current Context Zone](#) and the [Financial Information Zone](#) arrears are shown in a colorful bar (where each color corresponds to different aged buckets):



Whereas on [Service Agreement - Main](#), aged debt is shown in a grid:

Debt Class Arrears

Days Old	Arrears Amount
4	\$99.45
32	\$76.36

The grid method is used on many pages throughout the system. The following rows *may* appear in the grid:

- A row labeled New Charge highlights all debt that hasn't started aging yet. For example, if you've created a late payment charge and it hasn't appeared on one of the customer's bills, it will be classified as a New charge until the next bill is completed for the customer (unless a user overrides the late payment charge's arrears date by drilling into the financial transaction).
- A row with a label containing n Future (where n is the number of days) appears if there is "future debt". Future debt is very rare and can only exist if a debit financial transaction has a future arrears date. Financial transactions can receive a future arrears date if a bill is completed with a future date or if a user overrides a financial transaction's arrears date with a future date).
- A row that contains a number (and nothing else) represents debt that has started aging. The number is the age of the respective debt. In the above example, the customer has 1 day old debt, and debt that is more than 150 days old. Notice that the 150 day old debt is prefixed with a +. This means that the related debt is more than 150 days old. This age limit is controlled by a field on [Installation Options - CC](#) called "Oldest Bucket Age". This field limits the number of days the system will age debt. For example, if you set this field to 150, the system will never age an FT more than 150 days (and all debt that's older than 150 days will be classified as 150 day old debt). Also note, the aged debt bar that appears on [Current Context Zone](#) only ages debt a maximum of 60 days.
- A row with a label of Disputed appears if the account is an [open-item](#) customer and this customer has [disputed](#) financial transactions.

FASTPATH:

Refer to [Financial Transactions And Aged Debt](#) for more information.

GL Accounting Information

Be aware that if payoff amount is non-zero, the financial transaction has general ledger detail lines.

There are unusual financial transactions whose payoff amount is zero, but still affect the general ledger:

- Bill segments for company usage do not impact payoff amount (because your organization doesn't really owe itself anything). However, the GL is affected.
- Payment segments for charitable contributions (created when your customers contribute extra money to a charity) do not impact payoff amount. Why? Because payoff amount is never debited when a charitable contribution is billed (the customer doesn't truly owe you for this receivable). It's only when the customer pays the contribution that the GL is impacted (debit cash, credit charitable contribution payable).
- If the SA has a special role of Loan, the financial transaction algorithms supplied with the base package transfer the current amount between the long-term receivables and the short-term receivables in the GL. This allows the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). Refer to [Payoff Balance and Current Balance for Loans](#) for more information.

The effect on your GL is controlled by the financial transaction algorithm defined on your bill segment and payment segment types.

FASTPATH:

Refer to [The GL Interface](#) for how GL account information is interfaced to the general ledger.

A Complicated Example

The financial ramifications of a revolving charge account are predictable (if you're an accountant). The following table outlines the different financial events and their impact on the general ledger, arrearage history, and the amounts due (both current and payoff).

Event	GL Accounting	Arrearage Rule	Effect On Payoff Amt	Effect On Current Amt	Payoff Balance	Current Balance
Merchandise purchased	A/R 1000 Revenue <1000>	n/a (current amount is zero)	+1000	0	1000	0
Monthly bill	A/R 10 Int. Rev <10>	\$120 aged accordingly	+10	+120	1010	120
Payment received	Cash 120 A/R <120>	\$120 relieved accordingly	-120	-120	890	0

The following points describe the events in the above table:

- **Merchandise purchased.** When a customer purchases an air conditioner:
 - The system generates an adjustment to book the purchase.
 - The customer doesn't really think they owe the entire \$1,000 (because they've purchased it on credit), therefore no moneys are booked to current amount. However, if the customer wanted to cash out, they would owe your organization \$1,000, therefore the entire amount of the purchase is booked to payoff amount.
 - Because no money was booked to current amount, this event has no impact on the arrearage history.
- **Customer billed.** Monthly, the system calculates how much the client owes. In this example, interest is calculated to be \$10 and the minimum monthly payment is set at \$120.
 - The interest is posted to the GL, but principal isn't since it was booked when the merchandise was purchased.
 - The customer really thinks they owe the minimum payment amount, \$120. Therefore, current amount is affected. However, if the customer were to cash out, they would owe your organization \$1,000 + \$10 (the interest); therefore payoff amount is affected by only \$10.
 - Because current amount changed by \$120, arrearage history is affected accordingly.
- **Payment received.** With any luck, the client will pay the \$120 that was billed (note, they could obviously pay more).
 - The payment has a normal affect on the GL (debit cash, credit A/R).
 - The amount the customer thinks they owe decreases by \$120, therefore current amount is affected by the payment amount. And, if the customer was to cash out, they would owe the utility \$120 less, therefore payoff amount is affected by the payment amount.
 - Because current amount changed by \$120, arrearage history is affected accordingly.

Financial Transactions Created Between Bills

The following diagram illustrates how frozen financial transactions (FT's) accumulate between bills and are swept onto the next bill produced for the account (when the bill is completed). This example assumes

After the last bill is completed, the account's service will have no unbilled financial transactions

SA Type: Elec, Resid, Gen No unbilled financial transactions	SA Type: Gas, Resid, Gen No unbilled financial transactions
---	--

When an account is levied a late payment charge, FT's are created and linked to the account's service

SA Type: Elec, Resid, Gen ADJ Electric Service Cur Amt: \$10.00 Payoff Amt: \$10.00	SA Type: Gas, Resid, Gen ADJ Gas Service Cur Amt: \$5.00 Payoff Amt: \$5.00
--	--

When a payment is applied to the account's debt, two FT's are created and linked to the account's service

SA Type: Elec, Resid, Gen ADJ Electric Service Cur Amt: \$10.00 Payoff Amt: \$10.00	SA Type: Gas, Resid, Gen ADJ Gas Service Cur Amt: \$5.00 Payoff Amt: \$5.00
PAY Electric Service Cur Amt: \$32.00- Payoff Amt: \$32.00-	PAY Gas Service Cur Amt: \$68.00- Payoff Amt: \$68.00-

When a bill is generated for an account, two bill FT's are created and added to the account's service

SA Type: Elec, Resid, Gen ADJ Electric Service Cur Amt: \$10.00 Payoff Amt: \$10.00	SA Type: Gas, Resid, Gen ADJ Gas Service Cur Amt: \$5.00 Payoff Amt: \$5.00
PAY Electric Service Cur Amt: \$32.00- Payoff Amt: \$32.00-	PAY Gas Service Cur Amt: \$68.00- Payoff Amt: \$68.00-
BILL Electric Service Cur Amt: \$42.00 Payoff Amt: \$42.00	BILL Gas Service Cur Amt: \$73.50 Payoff Amt: \$73.50

When a bill is generated for account, it sweeps all unbilled FT's onto itself

Prior Balance	\$100.00
Bill Corrections	\$0.00
Payments	\$100.00
Adjustments	\$15.00
New Charges	\$115.50
Current Balance	\$130.50
...	

When any type of financial transaction is frozen, it impacts the related service agreement's **current and payoff balances**. If you do not want adjustments and bill segments to affect the customer's balance until they appear on the customer's next bill, refer to [Preventing SA Balances And The GL From Being Impacted Until Bill Completion](#).

Notice the balances in the financial summary of the above bill:

- The **Prior Balance** is the ending balance from the customer's prior bill.
- The **Bill Corrections** portion is blank. It contains a value if you cancel / rebill a bill segment that appeared on an earlier bill.
- The **Payments** portion shows payment financial transactions (both new payments and cancellations) that have been created since the last bill.
- The **Adjustments** portion shows adjustment financial transactions (both new adjustments and cancellations) that have been created since the last bill.
- The **New Charges** portion shows bill financial transactions that were created when the bill was created.
- The **Current Balance** is the total amount owed.

FASTPATH:

If you practice [Open Item Accounting](#), refer to [Open Item Versus Balance Forward Accounting](#) for more information about financial transactions and bills.

Financial Transactions And Aged Debt

The system keeps track of how old a service agreement's current balance is in order to determine if the customer is in arrears (and therefore credit and collections processing should start).

A financial transaction (FT) impacts the related service agreement's current and payoff balances the moment it is frozen. However, some types of frozen FTs have no impact on a customer's aged debt until the next bill is completed for the account associated with the service agreement.

As described in the previous section, a frozen financial transaction (FT) waits in limbo until the customer's next bill is produced. This limbo period could be several weeks if the customer is billed infrequently. When the customer's next bill is completed, all such frozen FT's are linked to the bill. It is important to stress the following in respect of these FT's:

- If the FT decreases the amount of debt, the customer's aged debt is affected immediately regardless of whether the FT appears on a bill.
- If the FT increases the amount of debt, the amount the customer owes from an aged debt perspective may or may not be affected by the FT. There is a switch on an FT called New Charge that controls the arrears behavior.
 - If this switch is on, the amount of debt will be reflected as a "new charge" when you look at the customer's aged debt. This amount will remain classified as a "new charge" until the FT is swept onto a bill. The moment the FT is swept onto the customer's bill, the debt starts aging. This logic exists because you probably don't want to start aging an FT until the customer has actually seen it.
 - If this switch is off, the date on which the FT starts aging must be defined in the Arrears Date field. The Arrears Date is used to compute how many days old the debt is.

NOTE:

Aged debt limitations. It's important to be aware that there's a field on [Installation Options - CC](#) called "Oldest Bucket Age" that limits the number of days old the system will age debt. For example, if you set this field to 360, the system will never age an FT more than 360 days (and all debt that's older than 360 days will be classified as 360 day old debt). Also note, the aged debt bar that appears on [Control Central - Account Information](#) only ages debt a maximum of 60 days.

FASTPATH:

If you practice [Open Item Accounting](#), refer to [Open Item Versus Balance Forward Accounting](#) for information about how open-item FT's affect aged debt.

Preventing SA Balances And The GL From Being Impacted Until Bill Completion

It's important to understand that when any type of financial transaction is frozen, the related service agreement's balance is affected. For example:

- When a payment is frozen, the customer's balance is reduced.
- When an adjustment is frozen, the customer's balance is impacted.
- When a bill segment is frozen, the customer's balance is increased (typically).

For payments, there is no issue. However, for bill segments and certain types of adjustments, you may NOT want the customer's balance to be impacted until the next bill is completed. Consider the following scenarios:

- Late payment charges:
 - You can setup the system to create a late payment charge (i.e., an adjustment) say 5 days after an unpaid bill is due.
 - If the related adjustment is frozen, the customer's balance will be impacted. However, its impact will not affect [aged debt](#) until the next bill is completed. In other words, the amount of the frozen adjustment segment will appear as a "New Charge" until the bill is completed.
- Batch billing:
 - If a customer has multiple service agreements, it's possible for one of the service agreements to have a bill segment that's in error and the other service agreement's bill segment to be error-free.
 - If this happens and you have setup the bill cycle schedule to freeze bill segments if they're error-free, then you could have one bill segment frozen and another in error.
 - The frozen bill segment will impact the customer's balance. However, its impact will not affect [aged debt](#) until the bill is completed (and a bill cannot be completed until all of its bill segments are error-free). In other words, the amount of the frozen bill segment will appear as a "New Charge" until the bill is completed.

WARNING:

We'd like to stress that while a frozen financial transaction impacts a customer's balance the moment it is frozen, the amount of the financial transaction appears as a "New Charge" when viewing a customer's [aged-debt](#). This amount will remain classified as a "New Charge" until the next bill is completed (i.e., the customer's debt doesn't start aging until the next bill is sent to the customer).

While [aged-debt](#) isn't impacted by frozen FT's, the general ledger is. This is because a financial transaction is marked for [interface](#) to the general ledger when it is frozen. This can be problematic if you have a long period between FT freeze and bill completion (you could impact the general ledger but not impact the customer's balance). If this is unacceptable, you can setup the system to not allow certain types of FT's to be frozen until the next bill is completed. This means that neither the customer's balance nor the general ledger will be impacted until bill completion time. To do this:

- Choose the Freeze At Bill Completion option on [Installation Options - Billing](#).
- Examine each of your [adjustment types](#). Select Freeze At Bill Completion for those that should not impact the customer's balance or the general ledger until the next bill is completed. Select Freeze At Will for those that should impact the customer's balance and the GL when they are frozen. Typically, the only adjustment types for which you'd choose Freeze At Will option are those that cause a customer's balance to be reduced, those that are used to refund money to a customer, and those that are created at bill completion. Adjustment types for adjustments created during bill completion (e.g., by a bill completion algorithm) must have their adjustment freeze option set to Freeze At Will. Otherwise (i.e., if the option is Freeze At Bill Completion) they will not be frozen until a subsequent bill is completed.

Please be aware of the following in respect of the Freeze At Bill Completion options:

- If you turn on Freeze At Bill Completion on [Installation Options - Billing](#):
 - Users will not be allowed to freeze bill segments online. This means that the freeze button will be disabled on [Bill - Main](#), [Bill - Bill Segments](#) and [Bill Segment - Main](#).
 - The Billing background process will not freeze bill segments until all segments on a bill are error free (and permission has been granted on the bill cycle schedule to complete bills).
 - Bill segments will exist in the freezable state until the bill is completed.
- If you turn on Freeze At Bill Completion on [Adjustment Type - Main](#):
 - Users will not be allowed to freeze adjustments of this type online. This means that the freeze button will be disabled on [Adjustment - Main](#).

- Background processes that create adjustments will not freeze this type of adjustment. Rather, the adjustments will be frozen when the next bill is completed.
 - Adjustments of this type will therefore exist in the freezable state until the next bill is completed.
-

NOTE:

Alerts highlight freezable FT's. Please be aware that messages appear in the [Account Information - Financial Information Zone](#) and in the [Dashboard - Financial Information Zone](#) to highlight the existence of freezable financial transactions.

Please be aware of the following in respect of the Freeze At Will options:

- If you turn on Freeze At Will on [Installation Options - Billing](#):
 - Users will be allowed to freeze bill segments online. This means that the freeze button will be enabled on [Bill - Main](#), [Bill - Bill Segments](#) and [Bill Segment - Main](#).
 - The Billing background process will freeze bill segments when the individual segment is error-free (and permission has been granted on the bill cycle schedule to freeze bill segments).
 - Bill segments will exist in the frozen state regardless of whether the bill is completed.
 - The frozen bill segment's FT will be interfaced to the GL when the interface next runs.
 - All adjustment types must be also be set to Freeze At Will (otherwise they wouldn't get frozen).
- If you turn on Freeze At Will on [Adjustment Type - Main](#):
 - Users will be allowed to freeze adjustments of this type online. This means that the freeze button will be enabled on [Adjustment - Main](#).
 - Background processes that create adjustments will freeze this type of adjustment.
 - Adjustments of this type will exist in the frozen state prior to bill completion.
 - The frozen adjustment's FT will be interfaced to the GL when the interface next runs.

Forcing The Freeze Date To Be Used As The Accounting Date

Every financial transaction references an accounting date. The accounting date controls the accounting period to which the financial transaction is booked as described below:

- Every financial transaction references an accounting date and a service agreement
- Every service agreement references a service agreement type
- Every service agreement type references a GL division
- Every GL division references an [accounting calendar](#)
- The accounting calendar contains the cross-reference between the accounting date specified on the financial transaction and the related accounting period in your general ledger

The accounting date is populated on financial transactions when they are initially generated. The following points describe the source of the accounting date:

- The user who creates or cancels a bill segment online defines the accounting date as part of the generation/cancel dialog (note, the current date defaults).
- Bill segments that are produced by the [BILLING](#) background process have their accounting date defined on the [bill cycle schedule](#) that caused the bill to be created.

- The user who creates or cancels an adjustment online defines the accounting date as part of the generation / cancel dialog (note, the current date defaults).
- Payments are unusual in that their financial transaction is only created when they are frozen (rather than when the payment is first distributed amongst the account's service agreements). At payment freeze time, the accounting date is set to the current date.

For payments, there is no issue because the accounting date is only populated on the financial transaction when a payment is frozen. However, for bill segments and adjustments, your business practice may dictate that the freeze date should be used as the accounting date rather than the original accounting date. Alternatively, your business practice may dictate that the accounting date that's originally stamped on bill segments / adjustments should be used (unless this associated period is closed at freeze time). It's really a question of the interpretation of the local accounting rules. After you've decided on your approach, populate the **Accounting Date Freeze Option** on [Installation Options - Billing](#) with one of the following values:

- Choose Always change if the accounting date on your financial transactions should be populated with the freeze date (i.e., the current date when the financial transaction is frozen).
- Choose Change if period is closed if the accounting date defined when the financial transaction is generated should be used (unless the associated accounting period is closed).

Please be aware of the following in respect of your choice:

- If you choose Always change:
 - When a user freezes a bill segment online, they will be prompted to supply an accounting date. The current date will default, but the user can override this value.
 - When a user freezes an adjustment online, they will be prompted to supply an accounting date. The current date will default, but the user can override this value.
 - The **BILLING** background process will use the current business date as the accounting date on bill segments that it freezes.
 - Also note, if you have chosen the Freeze At Bill Completion **Bill Segment Freeze Option** on the [installation record](#), bill segments and certain types of adjustments are frozen when a bill is completed. This means that the accounting date on the related financial transactions will be set to the completion date (because the completion date is the freeze date with this setting). Refer to [Preventing SA Balances And The GL From Being Impacted Until Completion](#) for more information.
- If you choose Change if period is closed:
 - When a user freezes a bill segment online, they will only be prompted to supply an accounting date if the related accounting period is closed (because the accounting period closes after the bill segment is generated but before it's frozen). The current date will default, but the user can override this date.
 - When a user freezes an adjustment online, they will only be prompted to supply an accounting date if the related accounting period is closed (because the accounting period closes after the adjustment is generated but before it's frozen). The current date will default, but the user can override this date.
 - The **BILLING** background process will use the accounting date defined on the related bill cycle schedule as the accounting date on the bill segments that it creates and freezes.

NOTE:

The above installation option only controls the final accounting date for GL recording purposes. Rate and bill factor value selection based on accounting date uses the date as initially determined.

How Late Payment Charges Get Calculated

Late payment charges are system-generated adjustments used to penalize a customer for late (or no) payments. This section describes how to set up the tables that control how and when late payment charges are generated. The following points describe how and when late payment charges are calculated.

- When a bill is completed, the system marks it with the date on which late payment charges will be calculated if the bill is not paid.
 - This date is calculated by adding grace days to the bill's due date. Grace days are defined on the account's [Customer Class / Division](#).
 - This date will be zero if the account's [Customer Class / Division](#) indicates the account is not eligible for late payment charge processing.
- The late payment charge background process (referred to by the batch ID of [LATEPYMT](#)) selects all bills on or after their late payment charge date.
 - For each such bill, the system determines if its account satisfies the late payment charge eligibility criteria defined on the account's [Customer Class / Division](#). The eligibility criteria are defined in an algorithm and can therefore be as flexible as required.
 - If an account is eligible for late payment charges, the system checks each of the account's service agreements to determine if it is eligible for late payment charges (as defined on the service agreement's [SA Type](#)).
 - If a service agreement is eligible for late payment charges, the system calls the SA type's late payment charge calculation algorithm. This algorithm should calculate the late payment charge amount, if applicable and return the calculated amount and an appropriate adjustment type to use. If this algorithm returns this information, an adjustment is generated to levy the late payment charge.

FASTPATH:

Refer to [Setting Up Customer Classes](#) for more information about how to set up an account's due days and grace period. Refer to [SA Type - Main Information](#) for more information about enabling late payment charges calculations for your service agreements.

You can update the **Late Payment Charge Details** section on the Bill - Main Information page to indicate if and when late payment charges may be levied. For more information, see [Bill - Main Information](#).

Service Agreement Type Controls Everything

The previous section illustrated three important concepts:

The true financial impact of the three financial events - bills, payments, adjustments - is at the service agreement level, not at the account level. This means that bills and payments are meaningless on their own. It's the service agreements' bill segments, payment segments and adjustments that affect how much a customer owes.

- Every bill segment, payment segment, and adjustment has a related financial transaction. These financial transactions contain the double-sided journal entries that will be interfaced to your general ledger. They also contain the information defining how the customer's debt is affected by the financial event (i.e., current amount and payoff amount).
- A single bill can contain many bill segments, each of which may have a different frequency. For example, a bill could contain future charges, monthly retroactive charges based on service cycle, quarterly charges that must end on a quarter-end boundary.

You control the financial effects of the various financial events using a single field on the service agreement. This field is called the service agreement (SA) Type. In this section, we describe many of the tables that must be set up before you can create a SA type.

NOTE:

Foreshadowing. You will notice that we don't explain how to set up SA types in this section. This is because SA type controls numerous aspects of a service agreement's behavior in addition to its financial behavior. The non-financial

aspects are discussed in later chapters. It's only after you have set up all of the control tables in this manual that you'll be able to finally define your SA types. Refer to [Setting Up SA Types](#) for more information.

WARNING:

Take the time to define how you will record the various financial events in your general ledger before you attempt to set up these control tables. If you have simple accounting needs, this setup process will be straightforward. However, if you sell many services and use sophisticated accounting, this setup process will require careful analysis.

Setting Up CIS Divisions

There are two types of Divisions referenced on a SA type: a CIS Division and a GL Division. This is a rather powerful structure, but it can be confusing.

- General Ledger divisions typically comprise individual entities (e.g., companies) in your general ledger. You must set up a GL division for each such entity. The GL division's sole purpose in the system is to define the accounting period associated with financial transactions linked to service agreements associated with the GL division (service agreements are associated with GL divisions via their SA type). The system cares about accounting periods in order to prevent a user from booking moneys to closed periods. It also uses accounting periods when it produces the flat file that contains the consolidated journal entry that is interfaced to your general ledger (refer to [The GL Interface](#) for more information).
- A CIS division is associated with a jurisdiction. The definition of a jurisdiction is a geographic-oriented entity with unique business rules. For example, if you conduct business in California and Nevada, and each state has different collection rules, you will need a separate jurisdiction for each state. You must set up a CIS division for each jurisdiction in which you conduct business.
- CIS division is also referenced on service agreement, premise and account.
 - The CIS division on SA is actually part of the SA's SA type. Because SA type controls many business rules, all business rules that are on the SA type can be thought of as being defined for a given jurisdiction and SA type combination. For example, you could define your valid rates for electric residential service in California which differ from the valid rates for electric residential service in Nevada. Refer to [Defining Service Agreement Types](#) for more information. In addition to controlling the business rules defined on the SA's SA type, the SA's CIS division also controls the type of collection criteria used to determine if and how to collect overdue debt. Refer to [Setting Up Collection Class Control](#) for more information.
 - The CIS division on premise defines the jurisdiction in which the premise is located. This jurisdiction controls the types of service agreements that can be associated with the premise's service points (e.g., you can only link California-oriented service agreements to premises governed by the California jurisdiction). You can also set up your field activity types to execute special algorithms when a field activity is completed at a service point located in specific jurisdiction.
 - The CIS division on account when combined with the account's customer class defines the jurisdiction that governs financial business rules (e.g., the bill's due date, when and how late payment charges are calculated, etc.). Refer to [Setting Up Customer Classes](#) for more information about these rules. The CIS division on account can also play a part in the addressee of To Do entries associated with the account. To assign To Do entries to a role based on the division, simply link the To Do type to the division. Refer to [To Do Entries Reference A Role](#) for more information.

NOTE:

Both CIS Division and GL Division are stored on the financial transactions associated with a service agreement. However, only GL Division plays a part in [The GL Interface](#). Refer to [SettingUp GL Divisions](#) for information about GL Divisions.

The topics in this section describe the pages used to maintain a CIS division.

CIS Division - Main

To define a CIS division, select **Admin > CIS Division > Add**.

Description of Page

Enter an easily recognizable **CIS Division** and **Description** for the CIS Division.

Enter the **Work Calendar** that defines the days on which this division operates. This calendar is used to ensure system-calculated dates (e.g., bill due date, credit and collection event dates, etc.) fall on a workday.

Use the **To Do Roles** scroll area if an account's division influences the role assigned to To Do entries associated with the account. In the collection, define the **To Do Role** to be assigned to entries of a given **To Do Type** that are associated with accounts that reference the **Division**. Refer to [Assigning A To Do Role](#) for more information.

NOTE:

Only To Do entries that are account-oriented take advantage of the roles defined for a division.

Where Used

Follow this link to view the tables that reference [CI_CIS_DIVISION](#) in the data dictionary schema viewer.

CIS Division - Characteristics

You can define characteristics for a CIS division. You may need these for reporting purposes or in your algorithms. Refer to [Characteristic Types](#) for more information.

Select **Admin > General > CIS Division > Search** and navigate to the **Characteristics** page to maintain a division's characteristics.

Description of Page

Select a **Characteristic Type** and **Characteristic Value** to be associated with this CIS division. Indicate the Effective Date of the characteristic type and value.

NOTE:

You can only choose characteristic types defined as permissible on a CIS division record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Setting Up Revenue Classes

Every service agreement references a service agreement (SA) type. Amongst other things, the SA type defines a service agreement's revenue class. The revenue class is used when the service agreement's rate books revenue to different GL distribution codes based on the service agreement's revenue class.

FASTPATH:

See [Designing Calculation Groups and Rules](#) for more information about how revenue class is used to determine the GL revenue accounts referenced on a bill. See [Revenue Segmentation](#) for more information about how revenue class affects the number of SA types you will need.

To set up revenue classes, choose **Admin > Financial > Revenue Class**.

Description of Page

Enter an easily recognizable **Revenue ClassID** and **Description** for every revenue class.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_REV_CL](#).

Setting Up Distribution Codes

Distribution codes simplify the process of generating accounting entries by defining valid combinations of chart of account field values.

FASTPATH:

Refer to [The Source Of GL Accounts On Financial Transactions](#) for more information about the accounting entries associated with bills, payments and adjustments.

To set up distribution codes, open **Admin > Financial > Distribution Code > Add**.

Description of Page

Enter a unique **Distribution Code** and **Description** for the distribution code.

If this distribution code is a holding account used for payables cash accounting, check the **Use For Non-Accrual Accounting** switch and select the accounting method from the **Accounting Method** list. Select the priority level for the distribution code from the **Accounting Priority** list and enter the actual payable **Accounting Code**. The system will transfer monies from the holding account to the distribution code when the cash event occurs. Transfers will occur based on priority and debt age. For more information, refer to [Payables Cash Accounting](#) and [Deferred Accrual Accounting](#).

Define the **GL Account Algorithm** used by the system when it interfaces financial transactions that reference this distribution code to your general ledger (refer to [GLDL - Create General Ledger Download](#) for more information about the download process). The logic embedded in this algorithm constructs the actual GL account number. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that constructs your general ledger account number. Click [here](#) to see the algorithm types available for this plug-in spot.

The **Write Off Controls** control how the system writes off debt associated with the distribution code. Refer to [The Ramifications of Write Offs in the General Ledger](#) for an explanation of how these fields are used at write-off time.

- Define the **Division** and **SA Type** of the service agreement to which bad debt associated with this distribution code should be transferred at write-off time. Note: only SA Types with a special role of Write Off may be selected.
- When the system transfers debt to the write-off service agreement defined above, the distribution code defined on this **Division / SA Type** will be debited unless you turn on the **Override Switch**. When this switch is turned on, the system overrides the distribution code of the transfer to side of the adjustment with the distribution code associated with the debt being written off. You'd typically turn this switch on for liability distribution codes because you want to debit the original liability account when the debt is written off. Note: if this switch is on the system also overrides the characteristic type / value with the respective value associated with the debt that is being written off.

Use the **GL Account Details** scroll to define how the system constructs the GL account associated with the distribution code when it interfaces the financial transaction to your general ledger. For each distribution code, enter the following information:

- Enter the **Effective Date** of the following information.
- Define whether, on the **Effective Date**, the following information is Active or Inactive. The system will only use effective-dated information that is Active.
- Enter the **GL Account** that the general ledger uses to process financial transactions tagged with this distribution code.

- Enter the **Statistics Code** that should be passed to the general ledger during the GL interface for this **GL Account**. For example, if this **Distribution Code** is used to record electric, residential revenue, the **Statistics Code** would be kWh if you record the number of kWh in your general ledger along with the dollar value of the revenue.
- If you have configured your installation options to indicate that [fund accounting](#) is practiced, define the **Fund** associated with this distribution code. If your installation options indicate that fund accounting is not practiced, the field is not visible.
- Use the grid to define characteristic values for the **Distribution Code**. To modify a characteristic, simply move to a field and change its value. The following fields display:
 - **Characteristic Type**. Indicate the type of characteristic.
 - **Characteristic Value**. Indicate the value of the characteristic.

NOTE:

You can only choose characteristic types defined as permissible on the distribution code record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_GL_DIVISION](#).

Setting Up Billable Charge Templates

A user creates a billable charge whenever a customer should be levied an ad hoc charge. For example, you would create a billable charge to charge a contractor for the repair of a ruptured gas line.

NOTE:

Interfacing billable charges from an external system. In addition to being entered manually, billable charges can also be interfaced from an external system. You would interface billable charges if your organization provides "pass through" billing services for a service provider. Refer to [Uploading Billable Charges](#) for more information.

A billable charge must reference a service agreement. This service agreement behaves just like any other service agreement:

- **Bill segments are created for the service agreement.** Whenever billing is performed for an account with billable charge service agreements, the system creates a bill segment for each unbilled billable charge.
- **Payments are distributed to the service agreement.** Payments made by an account are distributed to its billable charge service agreements just like any other service agreement.
- **Overdue debt is monitored.** The credit and collections process monitors billable charge service agreements for overdue debt and responds accordingly when overdue debt is detected.

NOTE:

Rates can be applied to billable charges. Billable charges can be connected to a service agreement that also specifies a rate. The rate will be applied and lines added to the bill segment after the billable charge lines are added. For example, a rate can insert flat charges or be applied to service quantities associated with the billable charge.

Taxes on top of billable charges. Rates cannot be applied to billable charge lines. If you need to perform a calculation such as applying taxes on top of the existing lines, add a service quantity (SQ) that contains the taxable amount with an SQ identifier that describes it as a taxable amount. A calculation rule can apply the tax to this SQ.

FASTPATH:

Refer to [How To Create A One-Time Invoice](#) for instructions describing how to create a bill for a billable charge outside of the normal bill creation process.

Billable charge templates exist to minimize the effort required to create a billable charge for a customer. A billable charge template contains the default bill lines, amounts and distribution codes used to levy a one-off charge.

The information on the template may be overridden by a user when the billable charge is created. For example, you can create a billable charge template to levy tree-trimming charges. This template would contain the bill lines, amounts and distribution codes associated with a tree trimming activities. Then, when you trim a tree for a customer, a user can create a billable charge using the template and override the amount to reflect the actual amount (if it differs from the norm).

NOTE:

Templates aren't required. A billable charge can be created without a template for a truly unexpected charge.

After setting up the billable charge templates, you must indicate the SA types that can use each template. Obviously, only billable charge SA types (as defined on the SA type's special role) will reference billable charge templates.

Billable Charge Template - Main

Open **Admin > Billing > Billable Charge Template > Add** to define your billable charge templates.

Description of Page

Enter a unique **Billable Charge Template ID** and **Description** for the billable charge template.

Use **Description on Bill** to define the verbiage that should print on the customer's bill above the billable charge's line item details.

Use **Currency Code** to define the currency in which the billable charge's amounts are expressed.

Use the grid to define the line item details associated with the billable charge (note, the **Total Line Amount** field is automatically calculated. It is the sum of the **Charge Amount** on each of the Line Sequence items). The following fields are required for each entry in the grid.

Sequence Line sequence controls the order in which the line items appear on the bill segment.

Description on Bill Specify the verbiage to print on the bill for the line item.

Charge Amount Specify the default amount to charge for the line item.

Show on Bill Turn this switch on if the line item should appear on the customer's printed bill. It would be very unusual for this switch to be off.

Appears in Summary Turn this switch on when the amount associated with this line also appears in a summary line.

Memo Only, No GL Turn this switch on when the amount associated with this line does not affect the GL (or the total amount owed by the customer).

Distribution Code Specify the default distribution code associated with this line item.

If you use the drill down button on the left most column in the grid, you will be taken to the Line Characteristics tab with the selected line displayed.

FASTPATH:

For more information about creating a billable charge, refer to [Maintaining Billable Charges](#). For more information about billing billable charges, refer to [How To Create A One-Time Invoice](#).

Billable Charge Template - Line Characteristics

Open **Admin > Billing > Billable Charge Template > Search** and navigate to the **Line Characteristics** page to define your billable charge templates line characteristics.

Description of Page

The **Line Sequence** scroll defines the billable charge template line to which you wish to assign characteristic values.

To modify billable charge template line characteristics, simply move to a field and change its value. To add characteristics, press + to insert a row and then fill in the information for each field. The following fields display:

Characteristic Type The type of characteristic.

Characteristic Value The value of the characteristic.

Billable Charge Template - SQ Details

Open **Admin > Billing > Billable Charge Template > Search > Search** and navigate to the **SQ Details** page to define your billable charge templates service quantities.

Description of Page

To modify a template's service quantity, simply move to a field and change its value. To add a new service quantity to the billable charge template, press the + button to insert a row and fill in the information for each field. The following fields display:

Sequence Specify the sequence number of the SQ.

UOM Select the unit of measure of this SQ. One or more of UOM, TOU, or SQ identifier must be selected.

TOU Select the time of use period.

SQ Identifier Select the SQ identifier.

Service Quantity Specify the number of units of this service quantity.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_B_CHG_TMPLT](#).

Designing and Defining Bill Segment Types

Every service agreement references a service agreement (SA) type. Amongst other things, the SA type references a bill segment type. The bill segment type controls how bill segments and their related financial transactions are created.

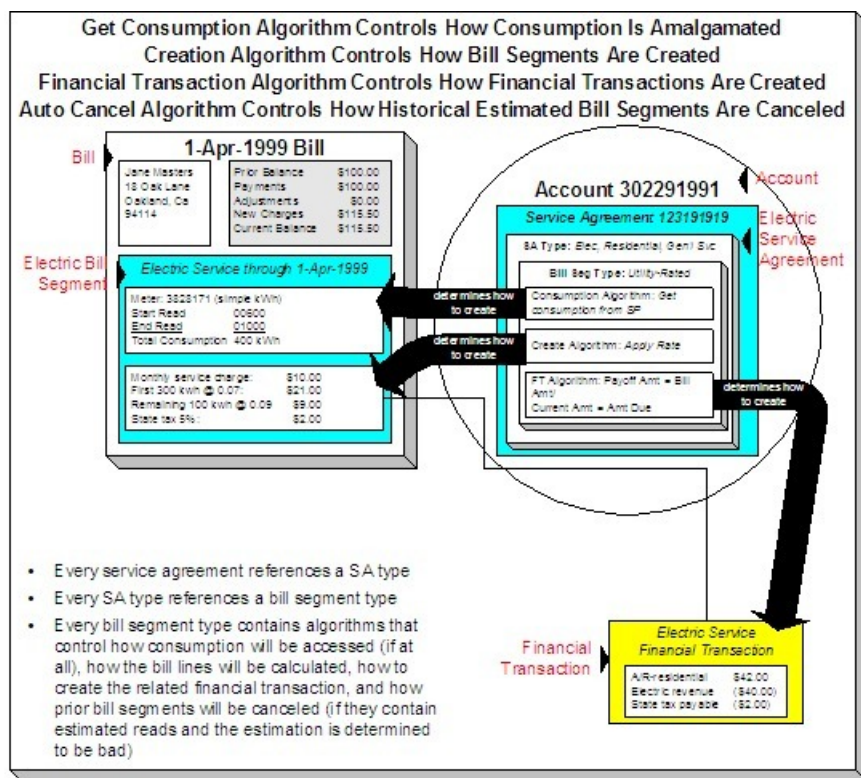
WARNING:

We strongly recommend understanding the concepts described in [The Big Picture of Billing](#) before setting up your bill segment types.

The topics in this section describe how to design and set up bill segment types.

What Do Bill Segment Types Do?

Bill segment types control how bill segments and their related financial transactions are created. The following illustration will help you understand how the system uses bill segment types during the bill segment creation process:



Designing Your Bill Segment Types

The following table contains a subset of the SA types listed under [Defining Service Agreement Types](#) and [Designing Your SA Types And Start Options For Sub SAs](#) and [Designing SA Types For Service Provider Financial Settlements](#). However, if you are reading this document from top to bottom, you probably don't know what your SA types are (they are only designed much later) and will have to forestall this task until that time.

We're going to cheat and assume you know what your SA types are and fill in the algorithms necessary to create bill segments for each SA type. After this table is complete, we will look for unique combinations of the 4 algorithms and create a bill segment type for each one.

NOTE:

Before you can fill in the columns for your own SA types, you should be comfortable with the descriptions of the algorithms described under [Setting Up Bill Segment Types](#).

Div-	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm
SA Type				
CA/G-RES	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	Auto cancel bad estimates
CA/G-COM	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption
CA/G-IND	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption

CA/CABLE	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption
CA/E-COY	Apply Rate	Payoff = 0 / Current = 0	Get Consumption From SP's	N/A - can't estimate consumption
CA/E-RESU	Apply Rate To Usage Request	Payoff = Bill Amount / Current = Amount Due	Get Consumption Using Usage Request	N/A - can't estimate consumption
CA/E-COMU	Apply Rate To Usage Request	Payoff = Bill Amount / Current = Amount Due	Get Consumption Using Usage Request	N/A - can't estimate consumption
CA/WO-STD	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable
CA/WO-LIA	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable
CA/CHARITY	Recurring Charge	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption
CA/PA-REGU	Recurring Charge With Auto Stop	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption
CA/MERCH-I	Recurring Charge With Auto Stop	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption
CA/DEP-I	Recurring Charge For Amount To Bill	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption
CA/ONETIME	Billable Charge	Payoff = Bill Amount / Current = Amount Due	N/A - no consumption is needed	N/A - no consumption
CA/OVR UNDR	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable
CA/E-SUB ENR	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From Master Bill Segment	N/A - rated sub service agreements are cancelled when there master is cancelled
CA/E-SUB BC	Billable Charge	Payoff = Bill Amount / Current = Amount Due	N/A - no consumption is needed	N/A - no consumption
CA/E-FIN SETTLE	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable

Now, we'll extract unique combinations of the 4 algorithms and create a bill segment type for each.

Bill Segment Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm
SP RATED	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	Auto cancel bad estimates
BD RATED	Apply Rate To Usage Request	Payoff = Bill Amount / Current = Amount Due	Get Consumption Using Usage Request	N/A - can't estimate consumption
NOESTRAT	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption
COMPUSAG	Apply Rate	Payoff = 0 / Current = 0	Get Consumption From SP's	N/A - can't estimate consumption
BILLCHRG	Billable Charge	Payoff = Bill Amount / Current = Amount Due	N/A - no consumption is needed	N/A - no consumption
RECUR	Recurring Charge	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption
RECUR AS	Recurring Charge With Auto Stop	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption

RECURATB	Recurring Charge For Amount To Bill	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption
SUB RATE	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From Master Bill Segment	N/A - rated sub service agreements are cancelled when there master is cancelled
SUB BC	Billable Charge	Payoff = Bill Amount / Current = Amount Due	N/A - no consumption is needed	N/A - no consumption

Just to make sure everything has been designed appropriately, we will return to our SA type samples and specify their respective bill segment types:

Div-SA Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm	Bill Segment Type
CA/G-RES	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	Auto cancel bad estimates	SP-RATED
CA/G-COM	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption	NOESTRAT
CA/E-RES	Apply Rate To Usage Request	Payoff = Bill Amount / Current = Amount Due	Get Consumption Using Usage Request	N/A - can't estimate consumption	BD RATED
CA/E-COM	Apply Rate To Usage Request	Payoff = Bill Amount / Current = Amount Due	Get Consumption Using Usage Request	N/A - can't estimate consumption	BD RATED
CA/G-IND	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption	NOESTRAT
CA/CABLE	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From SP's	N/A - can't estimate consumption	SP-RATED
CA/E-COY	Apply Rate	Payoff = 0 / Current = 0	Get Consumption From SP's	N/A - can't estimate consumption	COMPUSAG
CA/WO-STD	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	
CA/WO-LIA	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	
CA/CHARITY	Recurring Charge	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption	RECUR
CA/PA-REGU	Recurring Charge With Auto Stop	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption	RECUR-AS
CA/MERCH-I	Recurring Charge With Auto Stop	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption	RECUR-AS
CA/DEP-I	Recurring Charge For Amount To Bill	Payoff = 0 / Current = Bill Amount	N/A - no consumption is needed	N/A - no consumption	RECURATB

CA/ONETIME	Billable Charge	Payoff = Bill Amount / Current = Amount Due	N/A - no consumption is needed	N/A - no consumption	BILLCHRG
CA/OVR UNDR	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	
CA/E-SUB ENR	Apply Rate	Payoff = Bill Amount / Current = Amount Due	Get Consumption From Master Bill Segment	N/A - rated sub service agreements are cancelled when there master is cancelled	SUB RATE
CA/E-SUB BC	Billable Charge	Payoff = Bill Amount / Current = Amount Due	N/A - no consumption is needed	N/A - no consumption	SUB BC
CA/E-FIN SETTL	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	

And now you're ready to set up your bill segment types.

Setting Up Bill Segment Types

To set up bill segment types, open **Admin > Billing > Bill Segment Type > Add**.

Description of Page

Enter an easily recognizable **Bill Segment Type** and **Description** for every type of bill segment.

For each bill segment type, define the **Create Algorithm**. The logic embedded in this algorithm creates the bill segment. Refer to [Designing Your Bill Segment Types](#) for examples.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that creates a bill segment in the appropriate manner. Click [here](#) to see the algorithm types available for this plug-in spot.

WARNING:

The **BS Create Algorithm** is a very important field as it controls how the system creates bill segments. There are some restrictions in respect of the values of certain fields on the SA type and the bill segment algorithm used on a SA type. Refer to [Require Total Amount Switch versus Bill Segment Creation Algorithm](#), [Allow Recurring Charge Switch versus Bill Segment Creation Algorithm](#), and [Rate Required Switch versus Bill Segment Creation Algorithm](#) for more information.

For each bill segment type, define the **Financial Algorithm**. The logic embedded in this algorithm constructs the financial transaction associated with the bill segment. Refer to [Designing Your Bill Segment Types](#) for examples.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that constructs the bill segment financial transaction in the appropriate manner. Click [here](#) to see the algorithm types available for this plug-in spot.

FASTPATH:

For more information about current and payoff amounts, refer to [Current Amount versus Payoff Amount](#).

If the bill segment requires consumption (e.g., meter reads) to be retrieved, define the **Get Consum Algorithm**. The logic embedded in this algorithm retrieves the consumption that is billed on the bill segment. Refer to [Designing Your Bill Segment Types](#) for examples.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that retrieves consumption in the appropriate manner. Click [here](#) to see the algorithm types available for this plug-in spot.

The **Auto Cancel Algorithm** is used by the system when it detects that a prior bill segment contains an estimated read that meets certain conditions when a non-estimated read is used on the current bill. Examples include:

- The prior bill segment contains a bad estimated read (by "bad" we mean that the current bill has a non-estimated reading that is less than the estimated end read on the prior bill segment).
- The prior bill segment contains an estimated read (there may be situations where you want to cancel an estimated bill segment when a non-estimated read is also higher than the end read on the prior bill segment. For example, when tiered rates may cause a customer to be penalized by consumption being charged at a higher rate).

For detecting a bad estimated read on the prior bill segment, if you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that cancels bill segments that contain poorly estimated consumption. Click [here](#) to see the algorithm types available for this plug-in spot.

For detecting an estimated read on the prior bill segment, if you haven't done so already, you must set up this algorithm in the system and configure a feature configuration option. To do this:

- Create a new algorithm for the Auto Cancel Algorithm.
- On this algorithm, reference an Algorithm Type that cancels bill segments that contain estimated consumption. Click [here](#) to see the algorithm types available for this plug-in spot (refer to [Setting Up Algorithms](#)).
- Configure the **Always Call Auto Cancel** option type on the Financial Transactions Option feature configuration and set the option type value to 'Y'. For more information about Feature Configurations, see [Defining Feature Configurations](#).

The **Bill Segment Information Algorithm** is used by the system to format the bill segment information that appears throughout the system. If the information you'd like displayed differs for bill segment types, you must set up this algorithm in the system. To do this:

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BILL_SEG_TYP](#).

Designing and Defining Deposit Classes

If you bill for deposits, you must set up one or more deposit classes. If your company does not bill for deposits, you can skip this section.

FASTPATH:

We strongly recommend familiarizing yourself with the concepts described in [The Big Picture Of Deposits](#) before tackling the information in this section.

The topics in this section describe how to design and set up deposit classes.

What Do Deposit Classes Do?

A deposit class contains the business rules that govern:

- How and when deposit interest is calculated.
- How the recommended deposit amount is calculated.
- When a deposit will be automatically refunded to a customer.
- When the system will recommend a new or additional deposit.

When you link a deposit class to a SA type, you are indicating that the SA type's service agreements are governed by the deposit class' business rules.

In addition to linking a deposit class to the SA types used to bill for a deposit, you must also define a deposit class on SA types whose debt is covered by a deposit. Consider the following examples:

- Assume your company sells electricity, gas, and water; but deposits are only held only for electric service. In this situation, you'd need one deposit class - Electric - and you'd associate it with both the electric deposit SA type and the electric usage SA type(s) (the gas and water SA types would NOT reference a deposit class).
- If your company can apply a deposit to any type of debt, then you'd have just one deposit class - General Deposit. You'd link this deposit class to the deposit SA type, and to the other SA types whose debt is covered by the deposit.

NOTE:

Non-cash deposits. You can also use the system to manage non-cash deposits (e.g., letters of credit, surety bonds, 3rd party deposits). Non-cash deposits are held in respect of an account (and an account may have an unlimited number of non-cash deposits). Each non-cash deposit must reference a deposit class. Why? Because the system amalgamates cash and non-cash deposits when it determines if an account is holding an adequate deposit. Refer to [3rd Party Deposits](#) for more information.

Designing Your Deposit Classes

A deposit class contains the business rules that govern:

- How and when deposit interest is calculated.
- How the recommended deposit amount is calculated.
- When a deposit will be automatically refunded to a customer.
- When the system will recommend a new or additional deposit.

You will need multiple deposit classes if any of the above rules / conditions differ for different types of customers. For example, if residential customers use a different recommended deposit algorithm as compared to commercial customers, you'd need one deposit class for residential and another for commercial.

You will need additional deposit classes if your customers can have multiple deposits where each deposit is restricted to a specific type of debt. For example, if separate deposits are held for regulated and unregulated debt (and a customer could hold a combination of regulated and unregulated debt), you'd need one deposit class for regulated debt and another for unregulated debt.

We'll design deposit classes to satisfy the needs of a theoretical company to help you understand how to design your deposit classes. The following points describe the deposit requirements of our theoretical company:

- The recommended deposit amount is 2 times the average bill (averaged over the last 12 months). This is true regardless of the type of customer or debt.

- The system should automatically refund a deposit to a customer after:
 - The deposit has been held for at least 6 months; and
 - The account's credit rating is greater than the credit rating threshold defined on the installation record (i.e., the credit rating is no longer considered bad)
- This is true regardless of the type of customer or debt.
- Interest is calculated every 6 months. The interest rate is defined using a bill factor (refer to [Setting Up Bill Factors](#) for more information). This is true regardless of the type of customer or debt.
- When it's time to refund a refund a deposit, all outstanding debt will be paid off first. If any moneys remain, a check should be sent to the customer for the remainder. This is true regardless of the type of customer or debt.
- A customer could have both regulated and unregulated debt under a single account. When this happens, separate deposits will be held for each type of debt (where the regulated deposit can only be used to satisfy regulated debt and the unregulated deposit can only be used to satisfy unregulated debt).

You'd need the following deposit classes to satisfy the above requirement:

Deposit Class	Recommended Amount Rule	Auto Refund Condition	Interest Rules	Deposit Refund Method
Regulated	2 x Average Bill	Held for 6 months and credit rating is good	Simple interest every 6 months	Apply to outstanding debt first, refund remainder with a check
Unregulated	2 x Average Bill	Held for 6 months and credit rating is good	Simple interest every 6 months	Apply to outstanding debt first, refund remainder with a check

You may wonder why two deposit classes are needed when the rules are the same for both? Well, besides defining the applicable business rules for a deposit service agreement, a deposit class is defined on the SA types whose debt is covered by the deposit class' deposit. So, if you have two different types of debt where each type of debt can have its own deposit, you'd need at two deposit classes. Each deposit class would be associated with the service agreements that are being secured by the deposit.

Refer to [Setting Up Deposit Classes](#) for a description of the various algorithms defined in respect of a deposit class.

Setting Up Deposit Classes

In the previous section, [Designing Your Deposit Classes](#), we presented a case study that illustrated a mythical organization's deposit classes. In this section, we explain how to maintain your Deposit Classes.

Deposit Class - Main

To set up deposit classes, select **Admin > Credit & Collection > Deposit Class > Add**.

Description of Page

Enter an easily recognizable **Deposit Class** and **Description**.

Use **Refund Description on Bill** to define the information that appears on the bill segment produced when it's time to refund the customer's deposit.

The remaining information on this page is used by the various deposit-oriented processes.

Refer to [Deposit Class - Good Customer](#) for information about the **Good Customer Algorithm**.

Refer to [Deposit Class - Recommendation](#) for information about the **Recommendation Algorithm** and **Review Tolerance Percentage**.

Refer to [Deposit Class - Refund Method](#) for information about the **Refund Method Algorithm**.

Refer to [Deposit Class - Refund Criteria](#) for information about the **Refund Criteria Algorithm**.

Refer to [Deposit Class - Refund Interest](#) for information about the **Interest Refund Algorithm** and **Months Between Interest Refund**.

Refer to [Deposit Class - Review Method](#) for information about the **Review Method Algorithm**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DEP_CL](#).

Deposit Class - Good Customer

On [Deposit Class - Main](#) you must define the **Good Customer Algorithm** used by the system when it determines if a customer is considered good (the system recommends new / additional deposits for bad customers). Refer to [Deposit Review](#) for a description of the background process that reviews customers for adequate deposits.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines if a customer is considered good. Click [here](#) to see the algorithm types available for this plug-in spot.

Deposit Class - Recommendation

On [Deposit Class - Main](#) you must define the **Recommendation Algorithm** used by the system when it calculates a customer's recommended deposit amount.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that recommends deposits. Click [here](#) to see the algorithm types available for this plug-in spot.

The system uses the **Review Tolerance Percentage** to prevent the recommendation of small deposits by the Deposit Review background process. For example, if this field contains 10(%), the system would only recommend an additional deposit if the recommended amount was more than 10% of the existing deposit.

Deposit Class - Refund Method

On [Deposit Class - Main](#) you must define the **Refund Method Algorithm** used by the system when it refunds a deposit to the customer.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that refunds a deposit to a customer. Click [here](#) to see the algorithm types available for this plug-in spot.

Deposit Class - Refund Criteria

On [Deposit Class - Main](#) you must define the **Refund Criteria Algorithm** used by the system when it determines if it should automatically refund a deposit to a customer. Refer to [Deposit Review](#) for a description of the background process that reviews deposits for refunds.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines if a customer qualifies for a deposit refund. Click [here](#) to see the algorithm types available for this plug-in spot.

Deposit Class - Refund Interest

On [Deposit Class - Main](#) you must define the **Interest Refund Algorithm** to define how the system calculates interest and how it refunds the interest to the customer.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that calculates interest on a deposit. Click [here](#) to see the algorithm types available for this plug-in spot.

Interest will be automatically calculated every X months where X is defined in **Months Between Interest Refund**. Refer to [Deposit Interest](#) for a description of the background process that calculates interest on deposits. Also note that interest is calculated when a [deposit service agreement is stopped](#).

Deposit Class - Review Method

On [Deposit Class - Main](#) you must define the **Review Method Algorithm** used by the system to determine what action to take if the system recommends a deposit (or additional deposit) amount for an account. Refer to [Review Deposits](#) for a description of the background process that reviews deposits for refunds. The algorithm supplied with the base product highlights new deposits and deposit amounts on the [Deposit Review](#) page.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines the review method the system uses if it recommends a deposit or additional deposit be applied to an account. Click [here](#) to see the algorithm types available for this plug-in spot.

Setting Up Non-Cash Deposit Types

Non-cash deposit types are used to indicate the type of monetary instrument used for non-cash deposits. Refer to [Non-Cash Deposits](#) for more information.

To define your non-cash deposit types, select **Admin > Financial > Non-Cash Deposit Type**.

Description of Page

To modify a non-cash deposit type, move to a field and change its value.

To add a new non-cash deposit type, insert a row, then fill in the information for each field. The following fields display:

Non-Cash Deposit Type The unique identifier of the non-cash deposit type.

Description The description of the non-cash deposit type.

Review Before Expiration This switch indicates if the system will create a To Do entry when non-cash deposits of this type are close to expiration.

Third Party Deposit This switch indicates if the system requires a reference to a 3rd party's deposit service agreement for this type of non-cash deposit. Refer to [3rd Party Deposits](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_NCD_TYPE](#).

Setting Up Payment Segment Types

Every service agreement references a service agreement (SA) type. Amongst other things, the SA type references a payment segment type. The payment segment type controls how payment segments and their related financial transactions are created. To set up payment segment types, open **Admin > Payment Segment Type**.

Description of Page

Enter an easily recognizable **Payment Segment Type** and **Description** for every type of payment segment.

FASTPATH:

For more information about the source of the distribution codes on financial transactions, see [The Source Of GL Accounts On Financial Transactions](#).

For each payment segment type, define the **Payment Segment Fin Algorithm**. The logic embedded in this algorithm constructs the actual financial transaction associated with the payment segment. Refer to [Examples of Common Payment Segment Types](#) for examples of how algorithms are used on common payment segment types.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that constructs the payment segment financial transaction in the appropriate manner. Click [here](#) to see the algorithm types available for this plug-in spot.

FASTPATH:

For more information about current and payoff amount, see [Current Amount versus Payoff Amount](#).

Examples of Common Payment Segment Types

The following table shows several classic payment segment types used by many organizations:

Payment Segment Type	Payment Segment Financial Transaction Algorithm
Normal payment (if you practice accrual accounting). Refer to Accrual versus Cash Accounting for more information.	$\text{Payoff} = \text{Pay Amount} / \text{Current} = \text{Pay Amount}$ (no cash accounting)
Normal payment (if you practice cash accounting). Refer to Accrual versus Cash Accounting for more information.	$\text{Payoff} = \text{Pay Amount} / \text{Current} = \text{Pay Amount}$ (plus Cash Accounting)
Charity payment	$\text{Payoff} = 0 / \text{Current} = \text{Pay Amount}$ (the GL is affected)

Non-CIS Payments (When the FT is created, the distribution code and GL account to credit is retrieved from the pay). Refer to [Non CIS Payments](#) for more information

$\text{Payoff} = \text{Pay Amount} / \text{Current} = \text{Pay Amount (no cash accounting)}$

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PAY_SEG_TYPE](#).

Designing And Defining Adjustment Types

A service agreement's debt may be changed with an adjustment. Every adjustment must reference an adjustment type. The adjustment type contains a great deal of information that is defaulted onto the adjustment, including whether the adjustment amount is calculated. It also controls many business processes associated with the adjustment. The topics in this section describe how to design and set up adjustment types.

What Do Adjustment Types Do?

An adjustment type contains the business rules that govern how its adjustments are managed by the system. Please refer to [The Big Picture Of Adjustments](#) for a complete description of how adjustment types impact the lifecycle of adjustments.

Setting Up Adjustment Types

The topics in this section describe how to set up adjustment types.

NOTE:

When a new adjustment type is added. When you introduce a new adjustment type, you must update one or more adjustment profiles with the new adjustment type. Why? Because adjustment profiles define the adjustment types that may be levied on service agreements (adjustment profiles are defined on SA types). If you don't put the adjustment type on an adjustment profile, the adjustment type can't be used on any adjustment.

Adjustment Type - Main

To set up adjustment types, open **Admin > Adjustment Type > Add**.

Description of Page

Enter a unique **Adjustment Type ID** and **Description** for the adjustment type.

The **AdjustmentAmount Type** indicates whether the adjustment amount is calculated or not. Select **Calculated Amount** when you want to use a rate to perform calculations to generate the adjustment amount otherwise select **Non-Calculated Amount**. Refer to [Setting Up Calculated Adjustment Types](#) for more information about calculated adjustments.

Enter the **Distribution Code** that references the GL account associated with the adjustment. For example, if this adjustment type is used to levy a charge for a bad check, the distribution code would reference the revenue account to which you associate such revenue. Note, the offsetting distribution code is kept on the SA type.

NOTE:

Distribution Code for Calculated Adjustments. Depending on the algorithm used for the [calculated adjustment](#), the distribution code may come from the adjustment type or the calculation lines of the algorithm. If the adjustment's

calculation algorithm gets the distribution code from the calculation lines, you do not need to specify a distribution code on the adjustment type.

FASTPATH:

For more information about the source of the distribution codes on financial transactions, see [The Source Of GL Accounts On Financial Transactions](#).

Enter the **Currency Code** for adjustments of this type.

Turn on **Sync. Current Amount** if adjustments of this type exist to force a service agreement's current balance to equal its payoff balance. These types of adjustments are issued before a service agreement's funds are transferred to a write-off service agreement. If this switch is on, choose an **Adjustment Fin Algorithm** that does not impact payoff balance or the GL, but does affect the SA's current balance (refer to [ADJT-CA](#) for an example of such an algorithm).

Enter a **Default Amount** if an amount should be **defaulted** onto adjustments of this type.

FASTPATH:

For more information about current and payoff amounts, refer to [Current Amount versus Payoff Amount](#).

If the AP Adjustment should be recorded in respect of the customer's 1099 amounts, indicate the **A/P 1099 Flag**. This would typically be used on the adjustment used to credit the deposit service agreement with accrued interest. The values of this field are Interest and Miscellaneous. This type of adjustment would also have an **A/P Request Type Code** selected, as 1099 reporting is handled in A/P.

Turn on **Print By Default** if information about adjustments of this type should print on the account's next bill.

Choose an **A/P Request Type Code** if this adjustment is interfaced to accounts payable (i.e., it's used to send a refund check to a customer). Refer to [A/P Check Request](#) for more information.

The **Adjustment Freeze Option** defines when adjustments can be frozen and therefore when a service agreement's balance and the general ledger are affected by an adjustment. Refer to [Preventing SA Balances And The GL From Being Impacted Until Bill Completion](#) to understand the significance of this option. Also note, if the **installation option's** Bill Segment Freeze Option is Freeze At Will, this field is defaulted to Freeze At Will and cannot be changed.

WARNING:

Adjustment types for adjustments created during bill completion (e.g., by a bill completion algorithm) must have their adjustment freeze option set to Freeze At Will. Otherwise (i.e., if the option is Freeze At Bill Completion) they will not be frozen until a subsequent bill is completed.

If adjustments of this type require approval, define an **Approval Profile**. For more information, refer to [The Big Picture of Adjustment Approvals](#).

Enter the verbiage to appear on the printed bill in **Description on Bill**.

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all adjustments of this type. These can be used for reporting purposes or in your algorithms.

Adjustment Type - Adjustment Characteristics

To define characteristics that can be defined for adjustments of a given type, open **Admin > Adjustment Type > Search** and navigate to the **Adjustment Characteristics** page.

Description of Page

Use the **Adjustment Characteristics** collection to define characteristics that can be defined for adjustments of a given type. Turn on the **Required** switch if the **Characteristic Type** must be defined on adjustments of a given type. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

Adjustment Type - Algorithms

To define algorithms for adjustments, open **Admin > Financial > Adjustment Type > Search** and navigate to the **Algorithms** page.

Description of Page

The grid contains **Algorithms** that control important adjustment functions. If you haven't already done so, you must [set up the appropriate algorithms](#) in your system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Adjustment Cancellation	Optional	<p>When an adjustment is canceled an algorithm of this type may be called to do additional work.</p> <p>Refer to The Lifecycle Of An Adjustment for more information about canceling an adjustment.</p> <p>Click here to see the algorithm types available for this system event.</p>
Adjustment Freeze	Optional	<p>When an adjustment is frozen an algorithm of this type may be called to do additional work.</p> <p>Refer to The Lifecycle Of An Adjustment for more information about freezing an adjustment.</p> <p>Click here to see the algorithm types available for this system event.</p>
Adjustment Information	Optional	<p>We use the term "Adjustment information" to describe the basic information that appears throughout the system to describe an adjustment. The data that appears in "Adjustment information" is constructed using this algorithm.</p> <p>Plug an algorithm into this spot to override the "Adjustment information" algorithm on installation options or the system default "Adjustment information" if no such algorithm is defined on installation options.</p> <p>Click here to see the algorithm types available for this system event.</p>

Adj. Financial Transaction	Required	<p>Algorithms of this type are used to construct the actual financial transaction associated with the adjustment. The financial transaction controls the adjustment's affect on the service agreement's payoff and current balances. It also constructs the information that is eventually interfaced to your general ledger. Refer to Examples of Common Adjustment Types for examples of how algorithms are used on common adjustment types.</p> <p>Click here to see the algorithm types available for this system event.</p>
Default Adjustment Amount	Optional	<p>Algorithms of this type are used to default the adjustment amount. Refer to Default the Adjustment Amount for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Determine SA	Optional	<p>Algorithms of this type are used to find a service agreement for which the adjustment can be posted. This plug-in is used particularly during adjustment upload when a staging record does not identify the SA ID. Refer to Interfacing Adjustments From External Sources for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Resolve Suspense	Optional	<p>Algorithms of this type are used to automatically resolve adjustments that are in suspense. Refer to Suspense Adjustments for more information</p> <p>Click here to see the algorithm types available for this system event.</p>
Generate Adjustment	Optional	<p>Algorithms of this type are used to calculate the adjustment amount if an adjustment type indicates that the adjustment amount is calculated. Refer to Setting Up Calculated Adjustment Types for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Validate Adjustment	Optional	<p>Algorithms of this type are used to validate information for the adjustment after it is generated.</p> <p>Click here to see the algorithm types available for this system event.</p>

Setting Up Calculated Adjustment Types

You can use an algorithm to calculate an adjustment amount for example if you need to calculate tax on a base amount or calculate a non-sufficient funds charge based on the customer's credit rating. Because the base package algorithm calculates adjustment amounts by calling the rate application, calculated adjustments are sometimes referred to as ratable adjustments.

NOTE:

Ratable Adjustments Appear Deceptively Simple. But, they are not. Calculated adjustments that use the base package algorithm have all the power and flexibility (and complexity) of the rate application. Anything that you can do with a rate can be applied to a calculated adjustment. For examples that illustrate the flexibility of the rate application (and therefore calculated adjustments), refer to [Rate Examples](#).

Adjustment types that indicate they are calculated have a generate adjustment algorithm. The base package algorithm defines the rate schedule used to calculate the adjustment as well as any UOM, TOU or SQI parameters.

To set up calculated adjustment types using the base package generate adjustment algorithm type:

- Define the rate that performs the calculations, including the rate schedule, calculation groups, and calculation rules. Refer to [Rates](#) for information.

NOTE:

If you create your own Generate Adjustment algorithm type, you may not need to set up a rate that performs the calculations. It depends on the needs of your algorithm type.

- Create a Generate Adjustment algorithm (refer to [Setting Up Algorithms](#)) that references the base package algorithm type that generates calculated adjustments (see the table above).
- If you want the generation algorithm's calculation lines to provide the distribution codes when the adjustment is posted to the GL, create an Adjustment Financial Transaction algorithm (refer to [Setting Up Algorithms](#)) that references an algorithm type that creates the adjustment's financial transactions using the calculation lines. A parameter of the adjustment financial transaction algorithm determines whether the distribution codes are taken from the adjustment type (AT) or calculation lines (CL). The system comes supplied with several sample algorithm types that [create adjustment financial transactions](#).
- Create an adjustment type where the **Adjustment Amount Type** is Calculated Amount, the Generate Adjustment event references the generation algorithm created above, and the Adj. Financial Transaction event references the adjustment financial transaction algorithm created above.

Examples of Common Adjustment Types

The following table shows several classic adjustment types used by many organizations:

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
NSF	NSF revenue	Your NSF fee	Payoff = Current = Adj. Amt Refer to ADJT-NM for an example of such an algorithm type	Yes	No	No
LPC	Late payment charge revenue	N/A	Payoff = Current = Adj. Amt Refer to ADJT-NM for an	Yes	No	No

			example of such an algorithm type			
CONNECT	Connection charge revenue	Your connection charge	Payoff = Current = Adj. Amt Refer to ADJT-NM for an example of such an algorithm type	Yes	No	No
CUSTREL	Customer relationship expense	N/A	Payoff = Current = Adj. Amt Refer to ADJT-NM for an example of such an algorithm type	Yes	No	No
ADDCHARG	Misc Revenue	N/A	Payoff = Current = Adj. Amt Refer to ADJT-NM for an example of such an algorithm type	Yes	No	No
XFER	Balance transfer clearing	N/A	Payoff = Current = Adj. Amt Refer to ADJT-NM for an example of such an algorithm type	Yes	No	No
WO SYNC	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	No	No	No
REFUNDAP	A/P clearing	N/A	Payoff = Current = Adj. Amt Refer to ADJT-NM for an example of such an algorithm type	Yes	Yes	No
DPA FIX	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	Yes	No	No
CHARITFX	N/A	N/A	Payoff = 0 / Current = Adj. Amt	Yes	No	No

			Refer to ADJT-CA for an example of such an algorithm type			
BUDG ON	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	Yes	No	No
BUDG OFF	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	Yes	No	No
BUDG FIX	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	Yes	No	No
DEPOSREF	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	Yes	No	No
DEPOSINT	Interest expense	N/A	Payoff = Current = Adj. Amt. Refer to ADJT-NM for an example of such an algorithm type. Or Payoff Amt = Adj Amt / Current Amt = 0. Refer to ADJT-TA for an example of such an algorithm type. Use the first method if you want to have the interest reflected	Yes	No	Yes

as a credit balance on the customer's bill. Use the second method if you roll the interest amount into the customer's existing deposit on hand.

DEPFIXCR	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to ADJT-CA for an example of such an algorithm type	No	No	No
----------	-----	-----	--	----	----	----

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ADJ_TYPE](#).

Setting Up Adjustment Type Profiles

Adjustment type profiles categorize your adjustment types into logical groups. When you link a profile to a SA type, you limit the type of adjustments to be linked to the SA type's service agreements. The creation of adjustment profiles and their linkage to SA types prevents inappropriate adjustments from being linked to your service agreements. More than one adjustment type profile may be linked to a SA type.

For example, you can create an adjustment type profile called Miscellaneous Fees and link to it the miscellaneous fee adjustment types. Then, you would link this profile to those SA types that are allowed to levy such fees.

NOTE:

Bottom line. An adjustment can only be linked to a service agreement if its adjustment type is part of an adjustment type profile that is valid for the service agreement's SA type. If an adjustment type is not linked to a profile, it could never be levied.

To set up adjustment type profiles, open **Admin > Financial > Adjustment Type Profile > Add**.

Description of Page

Enter a unique **Adjustment Type Profile** and **Description** for the adjustment type profile.

Indicate the **Adjustment Types** that are part of the profile.

Examples Of Common Adjustment Profiles

The following table shows several classic adjustment profiles used by many organizations (we've displayed some attributes from the adjustment type in the following table to help make it more understandable):

Adjustment Profile	Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm
--------------------	-----------------	---------------------------	----------------	---------------------------------

FEES	NSF	NSF revenue	Your NSF fee	Payoff Amt = Adj Amt / Current Amt = Adj Amt
	LPC	Late payment charge revenue	Your LPC	Payoff Amt = Adj Amt / Current Amt = Adj Amt
	CONNECT	Connection charge revenue	Your connection charge	Payoff Amt = Adj Amt / Current Amt = Adj Amt
MISCEXP	CUSTREL	Customer relationship expense	N/A	Payoff Amt = Adj Amt / Current Amt = Adj Amt
XFER	TRANSBAL	Balance transfer clearing	N/A	Payoff Amt = Adj Amt / Current Amt = Adj Amt
REFUND	REFUND	A/P clearing	N/A	Payoff Amt = Adj Amt / Current Amt = Adj Amt
DPA	ADJCURR	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
	SYNCCURR	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
CHARITY	CHAR FIX	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
BUDGET	BUDG ON	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
	BUDG OFF	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
	BUDG FIX	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
	FIX PAY	Customer relationship expense	N/A	Payoff Amt = Adj Amt / Current Amt = 0
DEPOSIT	DEPOSBILL	N/A	Your standard deposit amount	Payoff Amt = 0 / Current Amt = Adj Amt
	DEPOSINT	Interest expense	N/A	Payoff Amt = Adj Amt / Current Amt = Adj Amt
	SYNCCURR	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt
	ADJCUR	N/A	N/A	Payoff Amt = 0 / Current Amt = Adj Amt

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ADJ_TYP_PROF](#).

The Big Picture Of Adjustment Approval

Some implementations require adjustments to be approved by one or more managers before they impact a customer's debt and the general ledger. For example, an adjustment used to rebate a credit balance may require managerial approval before the rebate is sent to the customer. The topics in this section describe how to set up the system to support the approval of adjustments.

Approval Is Controlled By Approval Profiles

An approval profile contains the rules that define if and how an adjustment is approved. If an adjustment type does not reference an approval profile, the related adjustments do not require third-party approval before they impact a customer's debt. If an adjustment type references an approval profile, the approval profile's approval hierarchy defines if the adjustment requires approval and who the authorized approvers are. For example, an approval profile can be configured with the following approval hierarchy:

- Adjustments < \$0 require approval by the "credit approvers role"
- Adjustments >= \$0 and <= \$10 do not require approval
- Adjustments > \$10 and <= \$100 require the approval of a user that belongs to the "level 1 approvers role"
- Adjustments > \$100 require two levels of approval: first a user that belongs to the "level 1 approvers role" must approve the adjustment; afterwards, the adjustment must be approved by a user that belongs to the "level 2 approvers role"

NOTE:

Transfer adjustments. The term "transfer adjustment" refers to two adjustments that are used to transfer moneys between two service agreements. The adjustment with the positive amount is considered to be the debit adjustment; the other adjustment is considered the credit adjustment. When a transfer adjustment requires approval, only one of the adjustments needs to be approved. You control whether the debit side or the credit side of a transfer adjustment is used to control the approval process when you set up the approval profile.

Approval Profiles Can Be Linked To Multiple Adjustment Types

Approval hierarchies are frequently the same for many adjustment types. The system allows an approval profile to be linked to multiple adjustment types to simplify the definition and maintenance of the rules over time.

Adjustments Created In Batch Are Not Approved

The system assumes that no approval is necessary for adjustments created by batch processes even those whose adjustment type references an approval profile.

Approval Inserts A Step Into An Adjustment's Lifecycle

[The Lifecycle Of An Adjustment](#) explains how an adjustment is transitioned from the Freezable state to the Frozen state when it should impact the general ledger and the customer's balance. If an adjustment's adjustment type references an approval profile, the user cannot freeze the adjustment directly. Rather, the user must submit the adjustment for approval when it's ready and only when the last applicable approver approves the adjustment will it become Frozen.

NOTE:

Freeze during bill completion. You can configure the system to only freeze certain types of adjustments when the next bill is completed for the adjustment's account. When the last approver approves such adjustments, they remain in the Freezable. When the next bill is completed for the account, these adjustment become Frozen. Such adjustments that have not been approved at the time of bill completion will remain in the Freezable state. Refer to [Preventing SA Balances And The GL From Being Impacted Until Bill Completion](#) for more information.

Approval Requests Manage And Audit The Approval Process

Users submit an adjustment for approval using a dedicated button on the [Adjustment](#) page. When an adjustment is submitted for approval, the system creates an "approval request". The approval request determines if the adjustment requires approval and, if so, the list of approvers. If the adjustment does not require approval, the approval request is updated to indicate such and the adjustment is Frozen immediately (if freezing is allowed prior to bill completion). If the adjustment requires approval, the approval request's state becomes Approval In Progress and the approver(s) are notified.

NOTE:

Approval submission logic is customizable. The previous paragraph describes how the base-package works when an adjustment is submitted for approval. This logic resides in an algorithm that's plugged in on the C1-AdjustmentApprovalProfile business object in the Determine Approval Requirements system event. Your implementation can change this logic by developing a new algorithm and plugging it into this business object. If your logic is meant to supersede the base-package algorithm, remember to inactivate the base-package algorithm by adding an appropriate inactivation option to this business object.

To Do Entries Are Created To Notify Approvers

When an approval request detects an adjustment requires approval, it notifies the first approver by creating a To Do entry. The To Do entry is created using the To Do type and To Do role defined on the approval profile. All users who belong to the approving To Do role can see the entry. When a user drills down on an adjustment approval To Do entry, the [Adjustments - Approval](#) portal is opened. This portal contains summary information about the adjustment and the approval history of the adjustment. This portal is also where the user approves or rejects the adjustment.

When the first user in the To Do role approves an adjustment, the To Do entry is Completed and the approval request's audit log is updated. If there are no higher levels of approval required, the adjustment is Frozen (if freezing is allowed prior to bill completion) and the approval request is moved to the Approved state. If there are higher levels of approval required, a new To Do entry is created to the next To Do role in the approval hierarchy.

NOTE:

To Do entries can create email. A To Do entry can be configured to create an email message for every user in the To Do role to inform the user(s) of new adjustments requiring their attention. Refer to [To Do Entries May Be Routed Out Of The System](#) for the details.

Monitoring and Escalating Approval Requests

The base-package is supplied with an algorithm that your implementation can use to monitor approval requests that have been waiting too long for approval. This algorithm can complete the current To Do entry and create a new one for a different role when the timeout threshold defined on the algorithm's parameters is exceeded. If you've configured the system to send email for approval, this algorithm can also send x reminder emails (where x is defined on the algorithm's parameters) before the approval request is escalated to the new To Do role. Refer to [C1-APR-TMOUT](#) for more information about this algorithm. If you plan to enable this functionality, plug-in your configured algorithm on the Approval In Progress state on the C1-AdjustmentApprovalRequest business object.

Rejecting Deletes The Adjustment

When an adjustment is being approved, anyone with access to the adjustment can reject it by using the [Adjustments - Approval](#) portal. Users other than the current approver are allowed to reject an adjustment to allow an "in process" an adjustment to be withdrawn.

When an adjustment is rejected, the following takes place:

- The user is prompted for a reject reason.
- The approval request's audit log is updated with the reject reason and the approval request is moved to the Rejected state.
- The adjustment is deleted.

Designing Your Approval Profiles

The following points describe a recommended design process:

- Create logical groups of adjustment types where each group has the same monetary hierarchy and approvers. An approval profile will be required for each of these groups.
- The number of To Do types (if any) that need to be created is dependent on how the adjustment approval To Do entries should be organized on To Do lists. For example, if all approval request To Do entries can appear in the same To Do list, you can use the base-package adjustment approval To Do type. However, if your organization prefers each approval profile's To Do entries to appear in a distinct To Do list, a separate To Do type will be needed for each list. Note that the base-package is supplied with a To Do type called **C1-ADAPP** that should be used as the basis for any new approval request To Do type.
- The number of To Do roles is dependent on who approves your adjustments. At a minimum, you will require a separate To Do role for each level in your approval profiles. Remember that every user in a To Do role will see its entries (and receive email if you've configured the system to do such).
- Refer to [Monitoring and Escalating Approval Requests](#) for how to configure the system to escalate approval requests that have been waiting too long.
- If your implementation requires email notification when an adjustment requires approval, the following setup is required:
 - Set up an outbound message type, external system, and message sender. Refer to [To Do Entries May Be Routed Out Of The System](#) for the details.
 - Every To Do type referenced on your approval profiles should be configured as follows:
- Define the **F1-TDEER** batch process as the To Do type's routing process
- Set up an algorithm that references the **C1-ADJAREQEM** algorithm type and plug it in the External Routing system event.

Exploring Adjustment Approval Data Relationships

Use the following links to open the application viewer where the physical tables and data relationships behind the approval functionality are documented:

- Click [C1-APPR PROF](#) to view the approval profile maintenance object's tables.
- Click [C1-APPR REQ](#) to view the approval request maintenance object's tables.

Implementing Other Approval Paradigms

The above sections describe how the base-package adjustment approval process works. Because adjustment approval has been implemented using the C1-AdjustmentApprovalProfile and the C1-AdjustmentApprovalRequest business objects,

your implementation can add additional business rules and change the approval user interface as required. Alternatively, if your implementation has a radically different approval process, you can create a different business objects with their own business rules. To learn how to do this, please enroll in the Configuration Tools training class.

Setting Up Approval Profiles

Approval profiles contain the rules that control how adjustments are approved. To set up an approval profile, open **Admin > Financial > Approval Profile > Add**.

FASTPATH:

Refer to [The Big Picture Of Adjustment Approval](#) for a detailed description of how approval profiles govern the adjustment approval process.

The topics in this section describe the base-package zones that appear on the Approval Profile portal.

Approval Profile List

The Approval Profile [List zone](#) lists every approval profile. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent approval profile.
- Click the Add link in the zone's title bar to add a new approval profile.

Approval Profile

The Approval Profile zone contains display-only information about an approval profile. This zone appears when an approval profile has been broadcast from the Approval Profile List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click the **Edit** button to start a business process that updates the approval profile.
- Click the **Delete** button to start a business process that deletes the approval profile.
- Click the **Duplicate** button to start a business process that duplicates the approval profile.

Please see the zone's help text for information about this zone's fields.

Approval Profile's Adjustment Types

The Approval Profile's Adjustment Types zone lists every [adjustment type](#) that is governed by this approval profile. This zone appears when there is at least one adjustment type governed by the approval profile displayed in the Approval Profile zone:

To add an adjustment type to this list:

- Navigate to the Adjustment Type page and display the desired adjustment type.
- Specify the governing approval profile and update the adjustment type.

To remove an adjustment type from this list:

- Navigate to the Adjustment Type page and display the desired adjustment type.
- Change or remove its approval profile and update the adjustment type.

Designing and Defining Budget Plans

If you allow your customers to pay a budget amount each month (as opposed to their actual bill amount), you must set up one or more budget plans. If your company does not offer budget billing options, you can skip this section.

The topics in this section describe how to design and set up budget plans.

The Financial Impact Of Budget Plans

The only difference between a customer who participates in budget billing and one who doesn't is that budget billing customer have bill segments where payoff amount differs from current amount. Why? Because the payoff amount is the actual amount of the bill. The current amount is the amount the customer is expected to pay (i.e., their budget amount).

Let's run through an example of a customer on a budget to illustrate a service agreement where these two balances are not the same. The values in the payoff balance and current balance columns reflect the amount due after the financial transaction has been applied:

Date	Financial Transaction	Payoff Balance	Current Balance
1-Jan-99	Bill: \$125, Budget \$150	125	150
15-Jan-99	Payment: \$150	-25	0
2-Feb-99	Bill: \$175, Budget \$150	150	150
14-Feb-99	Payment: \$150	0	0
3-Mar-99	Bill: \$200, Budget \$150	200	150
15-Mar-99	Payment: \$150	50	0

FASTPATH:

For more information about current and payoff amounts, refer to [Current Amount versus Payoff Amount](#).

What Do Budget Plans Do?

A budget plan contains the business rules that govern:

- How the recommended budget amount is calculated.
- When and how a customer on an ongoing budget plan will have their budget amount periodically tried up.
- The conditions under which the system will highlight an existing budget amount as being anomalous with the customer's current use patterns.

You may have different budget plans for different customer segments. For example, customers with large bills may have their budget amount recalculated every month, whereas small customers may have their budget amount only recalculated annually. You define which budget plans govern a customer's bills via a **budget plan on the customers' accounts**. An account's initial budget plan is defaulted from its customer class. You may override an account's budget plan at will.

Designing Your Budget Plans

FASTPATH:

Refer to [Budget Billing](#) for background information about budget billing.

A budget plan contains the business rules that govern:

- How the recommended budget amount is calculated.
- When and how a customer on an ongoing budget plan will have their budget amount periodically trued up.
- The conditions under which the system will highlight an existing budget amount as being anomalous with the customer's current use patterns.

You will need multiple budget plans if any of the above rules / conditions differ for different types of customers. For example, if residential customers use a different recommended budget algorithm as compared to commercial customers, you'd need one budget plan for residential and another for commercial.

We'll design budget plans to satisfy the needs of a theoretical company to help you understand how to design your budget plans. The following points describe the budget requirements of our theoretical company:

- The recommended budget amount is the last year's real bill amounts plus any existing debit/credit balance divided by 12. This is true regardless of the type of customer.
- The frequency of budget true up is monthly for commercial customers and annually for residential customers.
- The system should highlight when a residential customer's budget is more than 30% out of whack with what their budget amount would be if it was recalculated.
- The system should highlight when a commercial customer's budget is more than 20% out of whack with what their budget amount would be if it was recalculated.

You'd need the following budget plans to satisfy the above requirement:

Budget plan	Recommended Amount Algorithm	True Up Algorithm	Monitor Algorithm
Residential	Average Bill	True up every 12 months	Highlight when more than 30% out
Commercial	Average Bill	True up every month	Highlight when more than 20% out

Refer to the Page Controls under [Setting Up Budget Plans](#) for a description of the various algorithms defined in respect of a budget plan.

Setting Up Budget Plans

In the previous section, *Designing Your Budget Plans*, we presented a case study that illustrated a mythical organization's budget plans. In this section, we explain how to maintain your Budget Plans.

Budget Plan - Main

To set up budget plans, select **Admin > Budget Plan > Add**.

Description of Page

Enter an easily recognizable **Budget Plan** and **Description**.

The remaining information on this page is used by the various budget-oriented processes.

Refer to [Budget Plan - Calculation Algorithm](#) for information about the **Calculation Algorithm**.

Refer to [Budget Plan - Monitor Algorithm](#) for information about the **Monitor Algorithm**.

Refer to [Budget Plan - True Up Algorithm](#) for information about the **True Up Algorithm** and **Months for True Up**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BUD_PLAN](#).

Budget Plan - Calculation Algorithm

On [Budget Plan - Main](#) you must define the **Calculation Algorithm** used by the system when it calculates a customer's recommended budget amount.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that calculates recommended budget amounts. Click [here](#) to see the algorithm types available for this plug-in spot.

Budget Plan - Monitor Algorithm

On [Budget Plan - Main](#) you must define the **Monitor Algorithm** used by the [Budget Monitor](#) background process when it determines if a customer's budget plan is out-of-sync with their consumption patterns.

NOTE:

What happens? If the algorithm determines that a customer's budget plan is out-of-sync with its current recommended amount, an entry is added to the [Budget Review](#) page.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that highlights if a customer's current budget amount is out-of-sync with their consumption patterns. Click [here](#) to see the algorithm types available for this plug-in spot.

Budget Plan - True Up Algorithm

On [Budget Plan - Main](#) you must define the **True Up Algorithm** used by the [Budget True Up](#) background process when it periodically true's up a customer's budget.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that true's up budget amounts. Click [here](#) to see the algorithm types available for this system event.

The system will automatically true up a customer's budget amount every X months (X is defined in **Months for True Up**).

Tender Management

When a payment is received, a tender is created to record what was remitted (e.g., cash, check, credit card). The topics in this section describe control tables that must be set up in order to remit tenders.

FASTPATH:

We strongly recommend [Tender Management and Workstation Cashiering](#) before setting up the control tables described in this section.

Setting Up Tender Types

Tender types are used to indicate the method in which the tender was made. A unique **Tender Type** must exist for every type of tender that can be remitted. For example, if you allow cash, checks, direct debits from a checking account, and direct debits from a credit card to be tendered, you'd need the following tender types:

Tender Type	Description	Like Cash	Generate Auto Pay	Require External Source ID	Require Expiration Date	External Type
CASH	Cash	Yes	No	N/A	N/A	N/A
CHEC	Check	No	No	N/A	N/A	N/A
OVUN	Cash drawer - over/under	No	No	N/A	N/A	N/A
DDCH	Direct debit - checking	No	Yes	Yes	No	Checking withdrawal
CRED	Direct debit - credit card	No	Yes	No	Yes	Credit card withdrawal

Go to **Admin > Financial > Tender Type > Add** to define your tender types.

Description of Page

Enter a unique **Tender Type** and **Description** for the tender type.

Turn on the **Like Cash** switch if this tender type is cash or the equivalent of cash. This indicator controls if the system generates a warning if a cash-only account remits a tender other than cash. It is also used to generate a warning for online cashiers to turn in their tenders when the cash-like amount exceeds the maximum amount balance defined for the [tender source](#).

Turn on **Generate Auto Pay** if this type of tender causes an automatic payment request to be routed to a financial institution. For example, this switch will be on if this tender type is used for direct debits from a customer's checking account (because every tender of this type will have an automatic payment request created when the tender is created).

The following fields are only used for tender types associated with automatic payments:

External Type This field is used by the background process that creates the information that is interfaced to the automatic payment source. Specifically, it controls the record type associated with the different types of automatic payments that are routed to the automated clearinghouse (ACH).

NOTE:

The values for this field are customizable using the Lookup table. This field name is EXT_TYPE_FLG.

Require Ext. Src. ID This switch indicates if an Auto-Pay Source that references this type of tender must contain an External Source ID. The External Source ID is the unique identifier of the financial institution to which the automatic payment will be routed.

This switch is typically turned on for tender types associated with checking / saving direct debits. It is turned off for tender types associated with credit card debits (you don't need an external source for a credit card debit, you just need the credit card number).

Expiration Date Required Turn this switch on if an Auto-Pay Option that references an auto-pay source that references this type of tender must also contain an expiration date (e.g., automatic debit / credit cards).

Turn this switch off for tender types associated with checking / saving direct debits.

Tender Authorization Indicates that tenders of a particular type require authorization prior to being created.

Business Object If **Tender Authorization** has a value of Required, a **Business Object** must be specified for the tender type. The primary function of this **Business Object** is to manage the authorization of payment tenders.

FASTPATH:

For more information on authorizing credit card payments, refer to the Tender Type - Credit Card with Authorization business object.

Turn on **Allow Cash Back** if the system should automatically calculate a cash back amount when a tender is remitted for this tender type and the amount tender exceeds the amount being paid.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TENDER_TYPE](#).

Setting Up Tender Sources

A unique **Tender Source** must exist for every potential source of funds. For example,

- Every cashiering station will have a unique tender source.

NOTE:

If your organization accepts alternate currency payments online, then a tender source must exist for each currency code accepted at the cashiering station.

- Every lock box will have a unique tender source.
- Your remittance processor will have a unique tender source.
- If you allow customers to pay bills automatically (e.g., via EFT), you'll need a tender source for each institution to which you route automatic payment requests. For example, if you route automatic payment requests to the automated clearinghouse (ACH), you'll need a tender source for the ACH.

For example, if you have 3 lock boxes, 2 cash drawers at an area office A, 2 cash drawers at area office B, and a single remittance processor, you'd need the following tender sources:

Tender Source	Type	External Source ID (Lockbox ID)	Default Starting Balance	Currency Code	Suspense Service Agreement
CASH-A01	Cashiering	N/A	150.00	USD	N/A
CASH-A02	Cashiering	N/A	150.00	USD	N/A
CASH-B01	Cashiering	N/A	150.00	USD	N/A
CASH-B02	Cashiering	N/A	150.00	USD	N/A
LB-INDUS	Lockbox	112910-A	N/A	USD	9291019281
LB-COMM	Lockbox	938219-C	N/A	USD	4739837372
LB-RESID	Lockbox	372829-B	N/A	USD	1912910192
REMIT	Lockbox	N/A	N/A	USD	1920038437

To set up a tender source, select **Admin > Tender Source > Add**.

Description of Page

Enter an easily recognizable **Tender Source** and **Description** for the tender source.

Define the **TenderSource Type**. Valid values are: Ad Hoc, Auto Pay, Online Cashiering and Lockbox. The system uses this information to prevent tender controls from different sources from being included under the same deposit control. In other words, you can't mix ad hoc, automatic payment, cashiering and lockbox tenders under the same deposit control.

FASTPATH:

For more information, refer to [Maintaining Deposit Controls](#).

If the source is an external system (e.g., a lockbox or an automatic payment destination), use **External Source ID** to define the unique identifier of the source. The background process that interfaces tenders from this source uses this information to create the appropriate tender control when it interfaces payments from external sources.

If this source is a cash drawer, define the **Default Starting Balance**. This balance is defaulted onto new tender controls and may be overridden.

NOTE:

The tender type of the **Start Balance** is defined on the installation record.

If this source is a cash drawer, define the **Max Amount Balance**. When the amount of [cash-like](#) tenders in a cash drawer exceeds this balance, a warning is issued to remind the cashier to turn in some of the funds to a tender control.

Define the **Currency Code** of tenders linked to this source. All tenders in a source must be of the same currency.

If this tender source is associated with payments that are [interfaced from an external source](#) (e.g., a lockbox or a remittance processor), use Suspense **Service Agreement** to define the service agreement whose account will hold uploaded payments with an invalid account. Refer to [Payment Upload Error Segmentation](#) for more information about suspense service agreements. Also note, because the payment upload process simply books payments that reference invalid accounts to the account associated with this service agreement, this account should belong to a customer class with the appropriate payment distribution algorithms. This may entail creating a new customer class that will only be used on these "suspense accounts". This customer class would need the following algorithms:

- We'd recommend using a simple payment distribution algorithm like [PYDIST-PPRTY](#) (distribute payment based on SA type's payment priority and the age of the debt).
- We'd recommend using an overpayment distribution algorithm like [OVRPY-PPRTY](#) (distribute overpayment to highest priority SA type).

Define the **Bank Code** and **Bank Account** into which the tender source's moneys will be deposited. The bank account defines the distribution code used to build the GL details for the payment. Refer to [The Source of GL Accounts on Financial Transactions](#) for more information. Note that the bank code and bank account can later be overwritten when entering Tender Deposits on [Deposit Control](#).

If this tender source is associated with payments that are [interfaced from an external source](#), for example tender sources associated with Auto Pay and Lockbox **Tender Source Types**, the information is also used as follows:

- The [payment upload process](#) uses this information to populate the bank and bank account when it creates deposit control records for the tender controls it creates during the interface. Refer to [Managing Payments Interfaced From External Sources](#) for more information.
- The [automatic payment interface](#) uses this information to populate the bank and bank account when it creates deposit control records for the tender controls it creates during the interface.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TNDR_SRCE](#).

NOTE:

If your organization accepts alternate currency payments online, then a tender source must exist for each currency code accepted at the cashiering station. When a user adds a tender control the system attempts to default a tender source based on the currency of the deposit control and the tender source(s) define on the user's record.

FASTPATH:

Refer to [Alternate Currency Payments](#) for more information.

Automatic Payment Options

If your customers can pay their bills automatically (via direct debit or credit card debits), you'll need to set up the various control tables described in this section.

IMPORTANT:

Besides the tables described in this section, additional values must also be added to control tables defined under [Tender Management](#). Specifically, refer to [Setting Up Tender Types](#) and [Setting Up Tender Sources](#).

FASTPATH:

Refer to [Automatic Payments](#) for more information about how automatic payments are handled in the system.

Setting Up Auto-Pay Route Types

Auto Pay Route Types are used to control when and how automatic payment requests are routed to a financial institution, and when the general ledger is impacted. Select **Admin > Auto Pay Route Type** to define your route types.

Description of Page

To modify an auto pay route type, simply move to a field and change its value.

To add a new route type, press + to insert a row, then fill in the information for each field. The following fields display:

Route Type The unique identifier of the route type.

Description The description of the route type.

Tender Source The background process that routes automatic payment requests to a financial institution (e.g., the automated clearing house interface) will mark each automatic payment's associated tender with a tender control for audit and control purposes. The following points describe how this happens:

- When the system sees that it's time to send an automatic payment to a financial institution, it looks at the automatic payment's auto-pay source.
- Every auto-pay source references an auto-pay route type.
- Every auto-pay route type references a tender source.
- A **Tender Source** has a tender control for each group of tenders deposited / interfaced together one batch.
- The system marks each automatic payment's associated tender with the latest tender control for the **Tender Source**. The system will create a new tender control each time it routes automatic payments to the tender source. Refer to [Managing Payments Interfaced From External Sources](#) for more information about tender source and tender control.

Extract Batch Cd This field defines the background process that interfaces the automatic payment requests to the financial institution.

Autopay Date Calculation Alg This algorithm populates 3 dates associated with the automatic payment: 1) the date the automatic payment will be sent to the financial institution, 2) the date the general ledger will be impacted by the automatic payment, 3) the date of the payment.

If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that populates automatic payment dates. Click [here](#) to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_APAY_RT_TYPE](#).

Setting Up Auto-Pay Source Codes

A unique **Auto-Pay Source** must exist for every bank / credit card company / bill payment service that your customer's use as the source of the funds when they sign up for automatic payment. For example,

- Every bank will have a unique auto-pay source.
- Every credit card company will have a unique auto-pay source.

To set up an auto-pay source, select **Admin > Auto Pay Source Type > Add**.

Description of Page

Enter an easily recognizable **Auto Pay Source Code** and **Description** for the auto-pay source.

The **Source Name** is the name of the financial institution.

When the system creates an automatic payment request, it also creates an associated payment tender. This tender (like all tenders) must have a tender type. This field defines the **Tender Type** associated with this auto-pay source's tenders. Refer to [Setting Up Tender Types](#) for more information.

The **External Source ID** is the unique identifier of the financial institution to which the automatic payment will be routed (e.g., the bank routing ID of the bank). This field is typically blank on automatic payments routed to credit card companies because the credit card company doesn't have an external source ID (whereas direct debits from banks must have a bank routing number). Whether this field is required is controlled by the **Tender Type**.

The **Auto Pay Route Type** controls when and how automatic payment requests get routed to a financial institution. It also controls when the general ledger is impacted by the automatic payments financial transaction. Refer to [Setting Up Auto-Pay Route Types](#) for more information.

The **Work Calendar** defines the financial institution's workdays. This information is used to determine the date on which automatic payment requests will be sent to the financial institution. Refer to [Setting Up External Workday Calendars](#) for more information.

The **Validation Algorithm** defines how the system validates the customer's account ID at the financial institution. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that validates the customer's account ID at the financial institution. Click [here](#) to see the algorithm types available for this plug-in spot.

Refer to [Account - Auto Pay](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_APAY_SRC](#).

SEPA Payments

The topics in this section provide background information about the Single Euro Payments Area (SEPA) payment functionality.

NOTE: This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization is participating in SEPA payment transactions such as direct debit collection.

What Is SEPA?

SEPA (Single Euro Payments Area) is a European Union (EU) integration initiative that is aimed at streamlining processes that are related to cross-border payments. SEPA consists of the EU member states plus a few additional countries. In SEPA, customers can make electronic Euro payments within and across these countries under the same rights and obligations. SEPA payment services are based on global ISO (International Organization for Standardization) standards.

SEPA Direct Debit

The SEPA Direct Debit (SDD) scheme allows a creditor to collect money from the debtor, with prior approval from the debtor. The direct participants are the following:

- Creditor
- Creditor's bank
- Debtor
- Debtor's bank

The debtor and creditor must each hold an account with a payment service provider located within SEPA. The accounts may be in Euro or in any other SEPA currency. However, the transfer of funds between the debtor's bank and the creditor's bank always takes place in the Euro currency. The indirect participants are the following:

- Clearing and Settlement Mechanisms (CSMs) such as an automated clearinghouse
- Intermediary Banks that offer intermediary services to debtor banks and/or creditor banks

SEPA Direct Debit Mandate

The mandate is the consent and authorization that the debtor gives to the creditor, to allow the creditor to initiate direct debit collections. The creditor is responsible for storing the original mandates, together with any amendments relating to the mandate or information regarding its cancellation or lapse. The contents of this section describe how mandates are issued and canceled.

Issuing a Mandate

The creditor initiates the issuance of a mandate by sending the mandate form (either paper or electronic) to the debtor with the creditor information filled in. The creditor information includes the creditor's unique identifier as a SEPA creditor. An ISO standard specifies the structure of the creditor identification, which includes country code, a check digit, the creditor's business code and a country-specific identifier for the creditor. The creditor ensures that the mandate form contains the mandatory legal wording and the mandatory set of information, as specified in the standards. The debtor must ensure that the required information is supplied and that the mandate is signed, either in writing or electronically. The mandatory debtor information includes the debtor's international bank account number (IBAN) and bank identifier code (BIC). The creditor

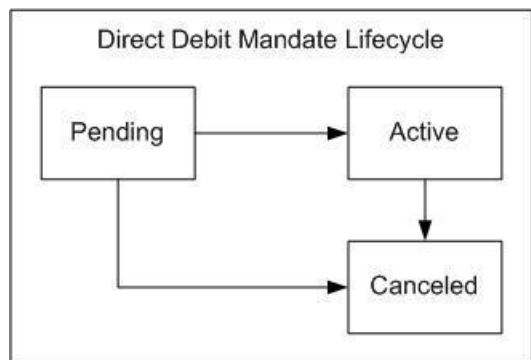
assigns a unique reference for the mandate and provides that reference to the debtor before the first initiation of a collection. The debtor can then use both the unique mandate reference and the creditor identification to verify the bank transactions.

Canceling a Mandate

A mandate can be canceled by either the debtor or the creditor, without the involvement of either bank.

The Lifecycle of a Direct Debit Mandate Task

The following diagram shows the possible lifecycle of the Direct Debit Mandate Task business object:



Pending State: The direct debit mandate starts out in a pending state.

Active State: The direct debit mandate is transitioned manually by a user to the active state. Once a mandate is created and activated in the system, the account's auto pay information must be updated to reflect information on the mandate. This process may be automated within the mandate's lifecycle and a sample algorithm exists in the demo database to achieve this. This algorithm is also included in the export bundle in the demo database.

Canceled State: The direct debit mandate is transitioned manually by a user to the canceled state.

Configuring the System for SEPA Direct Debits

The following sections describe the setup required if your organization intends to process SEPA direct debit transactions.

Define Auto-Pay Route Type

Configure an auto-pay route type for routing direct debit requests to a financial institution. This auto-pay route type must reference the SEPA Direct Debit Payment Extract (C1-SDDCE) batch process.

Refer to [Setting Up Auto-Pay Route Types](#) for more information.

Define Auto-Pay Source Type

Ensure that your auto-pay source codes reference the above auto-pay route type. You must also configure a validation algorithm on the auto-pay source type to ensure the International Bank Account Number (IBAN) defined on the Auto-pay Source of the Account has the correct format.

Refer to [Setting Up Auto-Pay Source Codes](#) for more information.

Define a Direct Debit Task Type

Your implementation must configure a **Direct Debit Task Type**. This service task type captures general information that the SEPA extract process needs to create the output file. The base product provides a business object for direct debit task type, **C1-DirectDebitMandateTaskType**.

Note that the direct debit task type is maintained on the Service Task Type portal.

Define SEPA Country Codes

The list of countries and territories that are part of the Single Euro Payment Area (SEPA) must be defined on extendable lookup **C1-SEPACountryCodeLookup**. These country codes are used by the system to validate the creditor identification entered on a direct debit task type.

Note that an export bundle exists in the demo database that includes a sample list of SEPA country codes. This list must be verified by your implementation for accuracy.

Define Mandate Cancel Reasons

A cancel reason is required when canceling a direct debit mandate. Ensure that these status reasons have been configured for the direct debit task business object, **C1-DirectDebitMandateTask**.

Define Navigation Options

The base product includes navigation options for easy access to query and maintain an account's direct debit mandate. To access these portals, add a new menu item to the account context menu (**CI_CONTEXTACCOUNT**). This menu item should include two menu lines:

- A context menu line for navigation to the direct debit mandate that references a navigation option of **c1DirectDebitMandate**. The system uses the account in context and attempts to find the most recent active mandate for the account. If a mandate exists, navigation is to the service task maintenance portal. If an active mandate does not exist, navigation is to the service task query portal.
- A context menu line for adding a new direct debit mandate that references a navigation option of **c1AddDirectDebitMandate**.

Setting Up Account Auto Pay

The SEPA direct debit extract process retrieves banking details from the account's auto pay record. Once a mandate is created and activated in the system, the account's auto pay information must be updated to reflect information on the mandate. This process may be automated within the mandate's lifecycle and a sample algorithm exists in the demo database to achieve this. This algorithm is also included in the export bundle in demo.

Payment Advices

The topics in this section provide background information about payment advice functionality.

NOTE:

This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization issues payment advices to the customer instead of initiating electronic funds transfer directly to the customer's bank.

What Is A Payment Advice?

Payment advice is a money order that is established at the initiative of the utility. When a bill is completed, the utility sends the customer a document that indicates a payment amount and the customer's bank details. If the customer agrees

to the information on the payment advice, he/she signs it and returns it to the clearinghouse address that is indicated on the payment advice. The clearinghouse, in turn, sends the dated and signed payment advice to the customer's bank, which completes the payment.

Payment Advice vs. Direct Debit

The existing functionality that creates automatic payments is referred to as direct debit processing. Payment advice processing differs from direct debit processing in the way that automatic payments get initiated. With payment advice processing, the usual automatic payment records - i.e. payment event, payment, tender and auto pay clearinghouse staging - are not created. Instead, a payment advice is printed and sent to the customer. The customer sends the approved payment advice directly to the clearinghouse.

NOTE:

The system does not provide sample processes for extracting and printing payment advice information. Your implementation team would have to create these.

Setting Up The System To Enable Payment Advices

You must set up a Financial Transaction Options [Feature Configuration](#) to define parameters that control payment advice functionality.

The following points describe the various **Option Types** that must be defined:

- Payment Advice Functionality Supported. This option controls whether the system allows for payment advice processing.
 - Enter Y if the system should allow for both direct debit and payment advice processing.
 - Enter N if the system should only allow for direct debit processing.
- Default Auto Pay Method. This option is used for defaulting the auto pay method on new account auto pay records.

Refer to [Account - Auto Pay](#) for more information on auto pay method.

NOTE:

The system assumes direct debit processing if the above feature options are not defined.

Credit Card Payments

If your organization accepts credit card payments, you can configure the system to authorize customers' credit card charges in real-time, and perform an authorization reversal (also in real-time) when the credit card payment is canceled. When the authorization web service is not available, you can permit users to enter authorization codes manually so that they can continue processing payments.

Configuring the System for Tender Authorization

The following sections describe the setup required if your organization intends to use the base CyberSource integration tender authorization functionality.

Define the Outbound Message Type

An outbound message type is required for the CyberSource authorization outbound message. This outbound message type must reference the base CyberSource - Credit Card Authorization business object.

An outbound message type is required for the CyberSource reversal outbound message. This outbound message type must reference the base CyberSource - Credit Card Reversal business object.

Define the Message Sender

A Message Sender is required to define how to send messages to CyberSource. Use the context of the Message Sender to define the web service interface.

Define the External System and Configure the Messages

Define an external system and configure the valid outbound message types and the method of communication for each (XAI, Batch, or Real Time ; Real Time is generally the appropriate choice for credit card authorization). You will also need to select the appropriate XSLs to format both the request and response to the outbound message types for CyberSource.

Define a User

Add a user to hold details required for CyberSource communication. Security information (e.g. Merchant Id, Merchant Reference Code, CyberSource User Name and Password) needed to interface with CyberSource is stored as user characteristics.

Set up the Tender Authorization Algorithm

A Tender Type (BO) - Tender Authorization algorithm must be configured. This algorithm performs a tender authorization or a tender authorization reversal through CyberSource.

Define a Business Object

A business object (BO) must be created for the TENDER TYPE maintenance object. This BO must reference the tender authorization algorithm created.

Define Tender Types

Update the appropriate tender type(s) to denote that authorization is required. The new business object must be specified on the tender type(s).

Tender Authorization - Feature Configuration

If your implementation has a need to prevent users from overriding the automatic tender authorization, then you must set up the Allow Manual Tender Authorization Override option type on the Financial Transaction Options [Feature Configuration](#). The Allow Manual Tender Authorization Override option must have a value of N in order to suppress the Authorization Override checkbox.

FASTPATH:

For more information on credit card payment authorization refer to the Tender Type - Credit Card with Authorization business object.

Non CIS Payments

Payment Templates can be configured for common types of non CIS payment allocations. These templates are used to default the payment distribution and allow non CIS payments to be directly allocated to specific distribution codes.

Setting Up Payment Templates

Payment templates contain the rules that control how non CIS payments are created. You can use a payment template to default the payment distribution for common types of non CIS payments. To set up a payment template, open **Admin** > **Payment Template** > **Add**. The topics in this section describe the base-package zones that appear on the Payment Template portal.

Payment Template List

The Payment Template [List zone](#) lists every payment template. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent payment template.
- Click the Add link in the zone's title bar to add a new payment template.

Payment Template

The Payment Template zone contains display-only information about a payment template. This zone appears when a payment template has been broadcast from the Payment Template List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click the **Edit** button to start a business process that updates the payment template.
- Click the **Delete** button to start a business process that deletes the payment template.
- Click the **Duplicate** button to start a business process that duplicates the payment template.
- Click the **Activate** or **Deactivate** button to start a business process that updates the status of the payment template.

Please see the zone's help text for information about this zone's fields.

Alternate Currency Payments

The topics in this section provide background information about alternate currency payments.

NOTE:

This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization accepts payments tendered in a currency other than the customer's currency. If your organization does not accept alternate currency payments, you need only indicate such on the [Installation Record](#); no other setup is required.

What Is An Alternate Currency Payment?

The currency code on the customer's account defines the currency in which the account's financial transactions are expressed. If the customer remits a payment in a different currency, this is referred to as an alternate currency payment. The system enables conversion of the tendered amount to the account's currency and captures the alternate currency and amount, as well as the exchange rate used in the conversion on the payment tender. The payment tender is linked to a tender control that references the alternate currency.

Configuring the System for Alternate Currency Payments

Allowing Alternate Currency Payments

You must set the **Alternate Currency** flag on the [Installation Record](#) to Allowed if your organization accepts alternate currency payments. This option controls whether the **Currency Converter** button is displayed when a payment is processed on the payment portal.

Payment Event Business Object

A business object (BO) must be created for the PAY EVENT maintenance object. You must specify this BO as the option value for the CIS Payment Event Add BO option type on the Financial Transaction Options [Feature Configuration](#). This BO must have the Currency Conversion Script BO option defined. This BPA script is invoked when the user clicks on the **Currency Converter** button during CIS payment processing on the payment portal.

NOTE:

Currency conversion logic is customizable. The base product includes a script for currency conversion called C1-ConvCurr that's plugged in on the C1-CISPaymentEvent business object. This script converts an alternate currency amount to the account's currency using a bill factor value. The bill factor to use is derived by concatenating the alternate currency code and the account's currency code. For example, if converting US Dollars (USD) to Barbados Dollars (BBD) the bill factor code to use would be USDBBD. Your implementation can change this logic by developing a new script and plugging it into the payment event business object.

Define the User's Tender Sources

Define the tender source(s) for the location (e.g., the specific cash drawer) in which a user's payment tenders are stored during the day. A tender source should be specified for each currency that payments are accepted in. The tender source(s) on the user record are used by the system when a user adds a new tender control. The system attempts to default a tender source on a new tender control based on the deposit control's currency and the tender source(s) defined on the user's record.

Payment Event Distribution

The base-package, by default, creates a single payment for a payment event. If your business requires potentially many payments to be created when payment events are added, you'll need to set up the various control tables described in this section.

FASTPATH:

Refer to [Distributing A Payment Event](#) for more information about how payment event distribution is handled in the system.

Making Payments Using Distribution Rules

As part of this method, one or more distribution details are provided at payment time along with the usual payment and tender information. Each distribution detail record references a distribution rule and a corresponding value. The distribution rule encapsulates the business rules that govern the distribution of the payment amount into payments using the specified value.

The type of value being captured on the distribution detail and the logic that uses it to create payments are defined on the [distribution rule](#).

Rule Value

The primary use of the rule value is to identify the business entity whose balance is to be relieved by creating payment(s). In most cases where the payor account is the same as the payee account it may also be used to identify the tender account associated with the payment(s).

Determine Tender Account

The very first step in processing a distribution detail is to identify the tender account (i.e. the payor) associated with the payment. To do that the system calls the Determine Tender Account [algorithm](#) defined on the distribution rule providing it with the rule value and other tender information.

Creating Payment(s)

The business logic that distributes a payment amount into one or more payments(s) targeted towards the entity identified by a rule value is held in designated Create Payment [algorithms](#) defined on the distribution rule.

Rule Value Can Capture Additional Information

A rule value can also be used to capture additional information provided at payment time, like address information, comments, etc. Obviously payment distribution details with this type of rule value should have a zero payment amount, as they are not real payments. These distribution details end up being linked to a payment event, but unlike other distribution details they do not contribute any payments. You can think of these details as payment event characteristics.

You don't have to set up a Create Payment algorithm for distribution rules intended solely to capture additional payment information.

Setting Up The System To Use Distribution Rules

Setting Up Distribution Rules

Define a Distribution Rule for each payment event distribution method practiced by your business.

Distribution Rule - Main

To set up a distribution rule, navigate to **Admin > Financial > Distribution Rule > Add**.

Description of Page

Enter a unique **Distribution Rule** and **Description** for the distribution rule.

Provide a short and unique **Distribution Rule Label** to be used as rule's name throughout the system.

Characteristic Type defines the type of entity whose balance is relieved by the payment(s) this rule creates. For example, if this rule targets payments(s) towards a specific service agreement, you'd reference a characteristic that its value identifies a service agreement. We use the term "rule value" for the characteristic value.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DST_RULE](#)

Distribution Rule - Algorithms

Navigate to **Admin > Financial > Distribution Rule > Search** and navigate to the **Algorithms** page to set up the algorithms appropriate for your distribution rule.

Description of Page

The **Algorithms** grid contains algorithms that control important functions. You must define the following for each algorithm:

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Optional / Required	Description
Create Payment	Optional	<p>This algorithm is executed to distribute a payment distribution detail payment amount into one or more payments.</p> <p>Click here to see the algorithm types available for this system event.</p>
Determine Tender Account	Optional	<p>This algorithm is executed to determine the tender account associated with the payment distribution detail.</p> <p>Only one such algorithm may be specified.</p> <p>Click here to see the algorithm types available for this system event.</p>

Feature Configuration

You must set up a Financial Transaction Options [Feature Configuration](#) to define parameters that control various payment event distribution options.

The following points describe the various **Option Types** that must be defined:

- Always Enable Distribution Rule. This option controls whether the system should only use the distribution rule method to add payment events or rather allow both the default method and the distribution rule method to coexist.
 - Enter Y if the system should always use distribution rules. With this setting, navigation to the Payment Event page in add mode opens up the [Payment Event Quick Add](#) page (defaulting it to the single payment event dialog). This dialog is designed to create a payment event using distribution rules
 - Enter N if the system should allow both methods. With this setting, navigation to the Payment Event page in add mode opens up the standard [Payment Event - Add Dialog](#) that uses the default method to create a payment event. If you want to use the distribution rule method, navigate to the Payment Event Quick Add page from the menu.
- Default Distribution Rule. This option states your default distribution rule that appears throughout the system.

Cancel Reasons

As described in [The Financial Big Picture](#), the various types of financial transactions can be canceled if their financial impact needs to be reversed from the system. Whenever a financial transaction is canceled, a cancel reason must be specified. This section describes the control tables that contain the cancel reason codes.

Setting Up Bill (Segment) Cancellation Reasons

Open **Admin > Billing > Bill Cancel Reason** to define your bill segment cancellation reason codes.

Description of Page

Enter an easily recognizable **Bill Cancel Reason** and **Description** for the bill cancellation reason.

Only use **System Default** on those reason codes that are placed on bill segments that are automatically canceled by the system. Valid values are: Turn off auto-cancel, Bad estimated read auto-cancel, MDM Corrected Read, and Mass Cancel. The reason code identified as Turn off auto-cancel is placed on bill segments that are automatically canceled when the final bill segment ends before the prior bill (and therefore we have to cancel the prior bill). The reason code identified as Bad estimated read auto-cancel is placed on bill segments that are automatically canceled by the system when it detects that it used an estimated read whose consumption is greater than the next actual read (and therefore we have to cancel the estimated bill segment). The reason code identified as Mass Cancel is placed on bill segments that are canceled as a result of the execution of the Mass Cancellation background process. The reason code identified as MDM Corrected Read is placed on bill segments that are automatically canceled by the system when processing a corrected read notification. Corrected read notifications are received from MDM when reads that were used in bill determinant calculation requests (usage requests) are replaced or modified. This notification may result in the rebill of frozen bill segments. Refer to [Usage Requests](#) for more information.

NOTE:

Required values. You must have one reason code defined for each of the System Default values.

Setting Up Payment Cancellation Reasons

Open **Admin > Financial > Pay Cancel Reason** to define your payment cancellation reason codes.

Description of Page

Enter an easily recognizable **Cancel Reason** and **Description** for the payment cancellation reason.

Turn on the **NSF Charge** switch if the system should invoke the non-sufficient funds (NSF) algorithm when a tender is cancelled using this reason code. Refer to [NSF Cancellations](#) for more information.

The next several fields are used to change an account's credit rating or cash-only points if a tender is canceled using the respective reason code.

- Use **Affect Cash-Only Score By** to define how tenders canceled using this reason will affect the account's cash-only score. This should be a positive number. When a customer's cash-only points exceed the cash-only threshold amount defined on the CIS installation record, the account is flagged as cash only during payment processing and on Control Central.
- Use **Affect Credit Rating By** to define how tenders canceled using this reason will affect the account's credit rating. This should be a negative number. A customer's credit rating is equal to the start credit rating amount defined on the CIS installation record plus the sum of credit rating demerits that are currently in effect.
- Use **Months Affecting Credit Rating** to define the length of time the demerit remains in effect. This information is used to define the effective period of the credit rating demerit record.

FASTPATH:

For more information, refer to [Account - Credit Rating](#).

NOTE:

The payor gets the credit rating / cash only hit. When you cancel a tender you must specify a cancellation reason. If the cancellation reason indicates a credit rating / cash only demerit should be generated, the system levies the credit rating transaction on the PAYOR's account.

The **System Default Flag** is specified on those cancellation reasons that are placed on payment segments that are automatically cancelled by the system. Valid values are: Re-opened Bill. The Re-opened Bill value is used as follows:

- Payments are automatically created for accounts who pay their bills automatically when their bills are completed.

- If such a bill is reopened before the automatic payment is interfaced to the paying authority, the system automatically cancels the payment. The Re-opened Bill cancellation reason is placed on such payments.

Setting Up Adjustment Cancellation Reasons

Open **Admin > Financial > Adjustment Cancel Reason** to define your adjustment cancellation reason codes.

Description of Page

Enter an easily recognizable **Cancel Reason** and **Description** for each adjustment cancellation reason.

Miscellaneous Financial Controls

This section describes miscellaneous control tables.

A/P Check Request

Adjustments whose adjustment type is marked with an A/P check request code are interfaced to your A/P system. Your A/P system then cuts the checks.

FASTPATH:

Refer to [Controls The Interface To A/P](#) for more information about the accounts payable interface.

You must set up at least one A/P check request code if you want A/P to cut checks.

To set up A/P check request types, open **Admin > A/P Request Type**.

Description of Page

Enter an easily recognizable **A/P Request Type** for the accounts payable request type.

Use **Due Days** to define when the check is cut. The cut date is equal to the adjustment date plus due days.

Select a **Payment Method**. Choose from these options:

System Check System check

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_APREQ_TYPE](#).

Billable Charge Line Type

NOTE:

Background information. Before using this page, you should be comfortable with the topics described under [Setting Up Billable Charge Templates](#) and [Uploading Billable Charges](#).

Billable charge line types will simplify the effort required to interface billable charges from an external system. Each line type contains values that will be defaulted onto the line details associated with the uploaded billable charges. Obviously, this defaulting is possible only if you specify a billable charge line type on the billable charge upload staging lines.

To set up billable charge line types, select **Admin > Billing > Billable Charge Line Type > Add**.

Description of Page

Enter an easily recognizable **Billable Charge Line External Type** and **Description**.

Use **Currency Code** to define the currency to be defaulted onto billable charge upload lines that reference this line type.

Use **Show on Bill** to define the value to be defaulted into the Show on Bill indicator on billable charge upload lines that reference this line type.

Use **App in Summary** to define the value to be defaulted into the App in Summary indicator on billable charge upload lines that reference this line type.

Use **Memo Only, No GL** to define the value to be defaulted into the Memo Only, No GL indicator on billable charge upload lines that reference this line type.

Use **Distribution Code** to define the values to be defaulted into the Distribution Code field on billable charge upload lines that reference this line type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_BCHG_UP_XTYP](#).

Payables Cash Accounting

In some areas, taxes and other 3rd party liabilities are not truly payable until the customer remits payment. We refer to this as "payables cash accounting". This practice should be contrasted with "payables accrual accounting" in which the liability is realized when the bill is created (as opposed to when it is paid).

NOTE:

Value Add Tax (VAT). VAT is a form of taxation common throughout the European Union. It is a common practice to only book the VAT payable when the customer remits payment. This means that most European implementations will use the functionality described in this section.

If your organization does not practice payable cash accounting, you may skip this section as accrual accounting is the system default. If you practice payables cash accounting, the contents of this section describe how to configure the system appropriately.

Accrual versus Cash Accounting Example

The following is an example of the financial events that transpire when a customer is billed and payment is received using accrual accounting.

Event	GL Accounting	Tax Payable Balance
Bill segment created	A/R 110	(10)
	Revenue <100>	
	Tax Payable <10>	
Payment received	Cash 110	(10)
	A/R <110>	

In the above example, you'll notice that the payable is booked when the bill is created. Let's contrast this with what takes place if the payable is subject to payables cash accounting.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
-------	---------------	---------------------	---------------------

Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Payment received	Cash 110	(10)	0
	A/R <110>		
	Tax Holding 10		
	Tax Payable <10>		

Notice that when the bill segment is produced, the liability is not booked, rather, the amount of the liability is placed in a "holding" GL account. When the customer pays, the moneys are transferred from the "holding" GL account to the true tax payable account.

NOTE:

Cash accounting is only applicable for liabilities. In the above example, you'll notice that only the tax payable account had cash accounting implications. This is because organizations that practice cash accounting only do it for liability accounts; they never do it for assets, revenue or expenses.

If the above seems simple, consider the following complications that must be considered:

- What happens if a partial payment is received?
- What happens if there are multiple taxes subject to cash accounting rules?
- What happens if the A/R is relieved via a deposit seizure (or transference of a credit balance from another SA)?
- What if, after payment, the original bill segment is cancel/rebilled resulting in a different amount of tax (keep in mind that the payable got booked when the payment was received)?
- What happens if the payment is cancelled?
- What if the payment isn't received and we have to write-off debt?
- What happens if the customer overpays?
- What happens if the customer is allowed to prepay their tax (this is a common practice in the United Kingdom) and then the tax rate changes at billing time?

The above points, and more, are discussed below.

Distribution Code Controls Cash Accounting For A GL Account

NOTE:

If you do not understand the significance of distribution codes, please refer to [Setting Up Distribution Codes](#).

Whether or not cash accounting is used for a specific GL account is defined on HOLDING GL account's distribution code (i.e., the holding GL account references the true payable account).

It is very important that unique payable and holding distribution codes be used for each type of tax subject to cash accounting rules. For example, if you have cash accounting requirements for both value-added tax (VAT) and a climate levy, you would need four distribution codes:

- VAT Payable.
- VAT Holding.
- Climate Levy Payable.

- Climate Levy Holding.

Without unique distribution codes for each payable and holding account, the system cannot keep track of how much of a given tax is being held, awaiting payment.

Bill Segments and Cash Accounting

The contents of this section describe how cash accounting is implemented when bill segments are created.

Bill Segment Financial Transactions Are Not Affected By Cash Accounting

There are NO changes to rate calculation associated with cash accounting. This is because the rate components that calculate tax reference the HOLDING payable distribution codes.

NOTE:

Prepaid taxes - future functionality. If your organization allows customers to prepay taxes in anticipation of a future tax increase (the customers receive the lower rate if they pay in advance), please speak to your account manager for information about when corresponding functionality will be available.

Payment Segments and Cash Accounting

The contents of this section describe how cash accounting is implemented when payment segments are created.

Payment Segment Financial Transaction Algorithms Transfer Holding Amounts to Payable GL Accounts

Logic exists in the pay segment's FT algorithm that transfers amounts from payable holding distribution codes to their respective payable real distribution codes.

FASTPATH:

Refer to [Setting Up Payment Segment Types](#) for how to define the appropriate FT algorithm.

The following table shows what happens to the financial transaction associated with the payment segment for a cash accounting customer.

Event	GL Accounting
Bill segment is created	A/R 110
	Revenue <100>
	Tax Holding <10>
Payment segment relieves receivables	Cash 110
	A/R <110>
Additional GL details created when the payment segment FT algorithm transfers the holding amount to a payable account	Tax Holding 10
	Tax Payable <10>
Net effect of the above	Cash 110

A/R <110>
Tax Holding 10
Tax Payable <10>

How Does The System Know What Amounts To Transfer From Holding To Payables?

When a payment segment is created for an account that is subject to cash accounting processing, the system determines if there is a CREDIT balance for any holding distribution code in respect of the service agreement. If so, it generates additional GL details to transfer moneys from the holding distribution code to the payable distribution code in proportion to the amount of receivables relieved by the payment. Therefore, if 100% of receivables are relieved by the payment segment, 100% of the holding amounts will be transferred to payable distribution codes. Refer to [Partial Payments Result In Partial Payables](#) for an example of what happens when a partial payment is created.

Partial Payments Result In Partial Payables

The previous example showed the entire tax holding amount being transferred to the tax payable account. The entire holding amount was transferred because the service agreement was paid in full. If a partial payment is received, only part of the holding amount will be transferred to the payable amount (proportional to the amount of receivables reduced by the payment). An example will help make the point.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Partial payment received	Cash 27.50	(2.50)	(7.50)
	A/R <27.50>		
	Tax Holding 2.50		
	Tax Payable <2.50>		

The above example assumes the use of the base product payment segment FT creation algorithm [PSEG-AC](#) to transfer the holding amount to the tax payable account. If multiple holding accounts are used, you may want to specify which holding amounts are relieved first. The base product includes an additional payment segment FT creation algorithm [C1-FTGL-PSAC](#) that handles booking holding amounts based on a priority.

Partial Payments Using Accounting Priority

To book holding amounts based on a priority, each holding distribution code must be assigned an **Accounting Priority**. When a partial payment is posted, only part of the holding amount will be transferred to the payable amount (proportional to the amount of receivables reduced by the payment). When the holding amount consists of various holding distribution codes with different accounting priorities, the amount to transfer is allocated as follows:

- Holding distribution codes associated with the oldest debt are settled first

- Within the same debt age, holding distribution codes with a higher accounting priority are booked first. If more than one distribution code shares the same priority, the settlement is distributed among them in proportion to the holding account balance

The above logic is handled by the payment segment FT creation algorithm [C1-FTGL-PSAC](#). As an example of how these rules apply, let's assume an implementation practices cash accounting; i.e. revenue, taxes and other third party liabilities are not recognized until payment is received. Also assume the following distribution codes have been configured:

Holding Distribution Code	Description	Cash Accounting Distribution Code	Accounting Priority
HLD-LPC	Late Payment Charge	R-MISC	10
HLD-RGEN	Revenue - Generation Charge	R-GEN	20
HLD-RDIS	Revenue - Distribution Charge	R-DIST	30
HLD-RTRN	Revenue - Transmission Charge	R-TRAN	30
HLD-THRD	3 rd Party Charges	R-THRD	40
HLD-VAT	VAT	A/P-VAT	90

Assume a customer has an outstanding third party charge with an arrears date of 2/Jan/2009:

Event	GL Accounting	SA's Payoff Balance	Holding Balances					
			HLD-LPC	HLD-RGEN	HLD-RDIS	HLD-RTRN	HLD-THRD	HLD-VAT
Unpaid Amount	A/R 50 HLD-HRD <45> HLD-VAT <5>	50	0	0	0	0	(45)	(5)

A bill is created for a customer and the result of the bill's financial transactions (an LPC adjustment in the amount of 10 and a bill segment in the amount of 127) include the following FT GL lines (both financial transactions have an arrears date of 15/Jan/2009):

Event	GL Accounting	SA's Payoff Balance	Holding Balances					
			HLD-LPC	HLD-RGEN	HLD-RDIS	HLD-RTRN	HLD-THRD	HLD-VAT
Bill segment created	A/R 127 HLD-RGEN <15> HLD-RDIS <20> HLD-RTRN <55> HLD-THRD <10> HLD-VAT <27>	177	0	(15)	(20)	(55)	(55)	(32)
Adjustment created	A/R 10 HLD-LPC <10>	187	(10)	(15)	(20)	(55)	(55)	(32)

No payment is received prior to the next bill. The result of the next bill's financial transaction (a bill segment in the amount of 100) includes the following FT GL lines (this financial transaction has an arrears date of 16/Feb/2009):

Event	GL Accounting	SA's Payoff Balance	Holding Balances					
			HLD-LPC	HLD-RGEN	HLD-RDIS	HLD-RTRN	HLD-THRD	HLD-VAT
Bill segment created	A/R 100	287	(10)	(30)	(40)	(100)	(65)	(42)
	HLD-RGEN <15>							
	HLD- RDIS <20>							
	HLD- RTRN <45>							
	HLD- THRD <10>							
	HLD-VAT <10>							

The following shows the result if a customer makes a payment on 20/Feb/2009. At payment time we'll build a table of holding amounts by accounting priority and debt age as follows:

Distribution Code & Priority	HLD-LPC	HLD-RGEN	HLD-RDIS	HLD- RTRN	HLD-THRD	HLD- VAT
	(10)	(20)	(30)	(30)	(40)	(90)
Debt Age						
4 days old		(15)	(20)	(45)	(10)	(10)
36 days old	(10)	(15)	(20)	(55)	(10)	(27)
49 days old					(45)	(5)

Examples of Partial Payments Using Accounting Priority

The examples below assume a customer has the financial history described above and attempts to illustrate the financial effect when a payment is made.

- [Example 1 - Customer Pays In Full](#)
- [Example 2 - Customer Makes a Partial Payment](#)
- [Example 3 - Customer Makes a Partial Payment](#)

Example 1 - Customer Pays In Full

Assume the customer makes a payment in the amount of 287. This amount is sufficient to satisfy all holding amounts, so the payment will have the following financial effect:

Event	GL Accounting	SA's Payoff Balance	Holding Balances					
			HLD-LPC	HLD-RGEN	HLD-RDIS	HLD-RTRN	HLD-THRD	HLD-VAT
Payment received	Cash 287	0	0	0	0	0	0	0
	A/R <287>							
	HLD-LPC 10							

R-MISC
 <10>
 HLD-RGEN
 30
 R-GEN <30>
 HLD- RDIS
 40
 R-DIST
 <40>
 HLD- RTRN
 100
 R-TRAN
 <100>
 HLD- THRD
 65
 R-THRD
 <65>
 HLD-VAT 42
 A/P-VAT
 <42>

[Top of the Page](#)

Example 2 - Customer Makes a Partial Payment

Assume the same financial history described above for a customer and a partial payment in the amount of 70 is made. This amount is not sufficient to satisfy the total holding amounts of 287, so the system will start settling held amounts starting with distribution codes with the oldest debt first from highest priority until the payment amount is exhausted.

A payment in the amount of 70 will be applied in the following sequence

Distribution Code & Priority	HLD-LPC (10)	HLD-RGEN (20)	HLD-RDIS (30)	HLD- RTRN (30)	HLD-THRD (40)	HLD- VAT (N/A)
Debt Age						
4 days old		(15.00)	(20.00)	(45.00)	(10.00)	(10.00)
36 days old	① (10.00)	② (15.00)	(20.00)	(55.00)	(10.00)	(27.00)
49 days old					③ (45.00)	④ (5.00)

The following describes how the holding amounts will be booked as a result of this partial payment:

- Settle oldest debt first (49 days old), i.e. 3rd Party Charges (HLD-THRD) and VAT (HLD-VAT). Note that even though these holding accounts have the lower accounting priorities, they are booked first because they have the oldest debt. An amount of 20 now remains on the partial payment.
- Next, we'll settle the 36 days old debt from the highest priority:
 - Late Payment Charge (HLD-LPC) in the amount of 10. An amount of 10 now remains on the partial payment
 - Revenue - Generation Charge (HLD-RGEN) gets the remaining payment amount of 10
- So, this partial payment in the amount of 70 will result in the following financial effect:

Event	GL Accounting	SA Balance	Holding Balances					
			HLD-LPC	HLD-RGEN	HLD-RDIS	HLD-RTRN	HLD-THRD	HLD-VAT
Payment received	Cash 70 A/R <70> HLD-LPC 10 R-MISC <10> HLD-RGEN 10 R-GEN <10> HLD- THRD 45 R-THRD <45> HLD-VAT 5 A/P-VAT <5>	217	0	(20)	(40)	(100)	(20)	(37)

[Top of the Page](#)

Example 3 - Customer Makes a Partial Payment

Assume the same financial history described above for a customer and a partial payment in the amount of 220 is made. This amount is not sufficient to satisfy the total holding amounts of 287, so the system will start settling held amounts starting with distribution codes with the oldest debt first from highest priority until the payment amount is exhausted.

A payment in the amount of 220 will be applied in the following sequence

Distribution Code & Priority	HLD-LPC (10)	HLD-RGEN (20)	HLD-RDIS (30)	HLD- RTRN (30)	HLD-THRD (40)	HLD- VAT (NA)
Debt Age						
4 days old		15 (15.00)	20 (20.00)	45 (45.00)	10 (10.00)	10 (10.00)
36 days old	10 (10.00)	15 (15.00)	20 (20.00)	55 (55.00)	10 (10.00)	27 (27.00)
49 days old					45 (45.00)	5 (5.00)

The following describes how the holding amounts will be booked as a result of this partial payment:

- Settle oldest debt first (49 days old), i.e. 3rd Party Charges (HLD-THRD) and VAT (HLD-VAT). Note that even though these holding accounts have the lower accounting priorities they are booked first because they have the oldest debt. An amount of 170 now remains on the partial payment.
- Next, we'll settle the 36 days old debt from the highest priority, i.e. Late Payment Charge (HLD-LPC), Revenue - Generation Charge (HLD-RGEN), Revenue - Distribution Charge (HLD-RDIS), Revenue - Transmission Charge (HLD-RTRN), 3rd Party Charges (HLD-THRD) and VAT (HLD-VAT). An amount of 33 now remains on the partial payment.
- Next we'll settle the 4 day old debt from the highest priority:
 - Revenue - Generation Charge (HLD-RGEN) in the amount of 15. An amount of 18 now remains on the partial payment
 - The two holding accounts at the next priority have an outstanding amount of 65. Since the remainder of the payment is not enough to satisfy this amount, the remainder of the payment is prorated amongst HLD-RDIS and HLD-RTRN as follows:

- (Remaining Pay Amount / Total Outstanding Holding Amount) * Holding Account Amount
- So for the Revenue - Distribution Charge (HLD-RDIS) holding account the amount booked will be $(18/65 * 20) = 5.54$
- So, this partial payment in the amount of 220 will result in the following financial effect:

Event	GL Accounting	SA Balance	Holding Balances					
			HLD-LPC	HLD-RGEN	HLD-RDIS	HLD-RTRN	HLD-THRD	HLD-VAT
Payment received	Cash 220	67	0	0	(14.46)	(32.54)	(10)	(10)
	A/R <220>							
	HLD-LPC 10							
	R-MISC <10>							
	HLD-RGEN 30							
	R-GEN <30>							
	HLD-RDIS 25.54							
	R-DIST <25.54>							
	HLD-RTRN 67.46							
	R-TRAN <67.46>							
	HLD-THRD 55							
	R-THRD <55>							
	HLD-VAT32							
	A/P-VAT <32>							

Adjustments That Behave Like Payments

There are several types of adjustments that behave just like payments (in respect of payables cash accounting). Consider the following events:

- Seizing a deposit (i.e., transferring a credit from a deposit service agreement to a regular service agreement)
- Overpayments transferred from one service agreement to another

The above events should cause the system to transfer holding amounts to true payable amounts (notice that the above examples are all transfer adjustments).

However, there are many other adjustments that should NOT behave like payments. You control how the adjustment works by selecting the appropriate FT algorithm when you [set up adjustment types](#) (refer to [ADJT-AC](#) and [ADJT-TC](#) for a description of the base package algorithms that cause the holding amounts to be manipulated in proportion to the amount of receivable being adjusted; and to [CI-FTGL-ADAC](#) and [CI-FTGL-ADTC](#) for the base package algorithms that take **Accounting Priority** into consideration). In other words, there are adjustment FT algorithms that cause the transference of holding payable amounts to real payable amounts when the A/R balance is decreased by the adjustment.

NOTE:

Cash refunds can behave like "anti-payments". In addition to the above examples of transfer adjustments behaving like payments, you should be aware that cash refunds may impact your holding and true payable balances. Refer to [Cash Refunds](#) for more information.

Overpayment Of Taxes Due To Cancel/Rebills

Let's assume a cancel / rebill occurs after a payment is received and the net effect of the cancel / rebill is that the customer has overpaid their taxes.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Payment received	Cash 110	(10)	0
	A/R <110>		
	Tax Holding 10		
	Tax Payable <10>		
Cancel	A/R <110>	(10)	10
	Revenue 100		
	Tax Holding 10		
Rebill	A/R 27.50	(10)	7.50
	Revenue <25>		
	Tax Holding <2.50>		

You'll notice that the amount payable to the taxing authority still indicates \$10 (the amount of tax that was paid by the customer). However, you'll notice that the tax holding balance is 7.50 (debit). This looks a bit odd, but it's correct. Remember that at this point, the customer has a credit balance of \$75 and this will be whittled down as successive bills are produced as shown below. Note: refer to [Cash Refunds](#) for an example of what happens if you refund the credit with a check rather than letting it whittle down.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
		(10)	7.50
Bill segment created	A/R 55	(10)	2.50

	Revenue <50>		
	Tax Holding <5>		
Bill segment created	A/R 110	(10)	(7.50)
	Revenue <100>		
	Tax Holding <10>		

In the unlikely event of a payment being received while the tax holding has a debit balance, nothing will be done in respect of transferring funds from holding to payable (there is nothing to transfer).

Cash Refunds

If you refund moneys to a cash accounting customer, it's important to do the opposite of what was done when the payment was received (i.e., you need to transfer the payable back to the holding account). The following example should help clarify this situation (this example shows a refund due to a credit balance that occurred as a result of a cancel/rebill).

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance	SA's Payoff Balance
Bill segment created	A/R 110	0	(10)	110
	Revenue <100>			
	Tax Holding <10>			
Payment received	Cash 110	(10)	0	0
	A/R <110>			
	Tax Holding 10			
	Tax Payable <10>			
Cancel	A/R <110>	(10)	10	(110)
	Revenue 100			
	Tax Holding 10			
Rebill	A/R 27.50	(10)	7.50	(82.50)
	Revenue <25>			
	Tax Holding <2.50>			
Payment refunded (via an A/P adjustment)	Cash <82.50>	(2.50)	0	0
	A/R 82.50			
	Tax Holding <7.50>			
	Tax Payable 7.50			

We understand this is tricky, but consider this - when a cash accounting customer makes a payment, the system transfers tax holding CREDIT balances to tax payable distribution codes in proportion to the amount of the receivable DEBIT amount that was reduced by the payment. Therefore, when cash is returned to the customer, the system should transfer tax holding DEBIT balances to tax payable distribution codes in proportion to the amount of the receivable CREDIT that was reduced by the refund.

NOTE:

The above takes place when an A/P adjustment is created if the related adjustment type references the appropriate FT algorithm (refer to [ADJT-AC](#) and [C1-FTGL-ADAC](#) for a description of the adjustment FT algorithms used for adjustments that behave like payments).

Overpayments

An overpayment, by definition, does not "match" to open items. However, the match type algorithms supplied with the base package will result in a balanced match event if an overpayment is made. The following points explain how this is achieved:

- The base package's match type algorithms will distribute the payment until the customer's current debt is satisfied.
- The amount of the overpayment will be kept on a separate SA (this only happens if you plug-in the appropriate Overpayment Distribution algorithm on your customer classes). Refer to [Overpayment Segmentation](#) for more information.
- When the payment is frozen, the payment segments that satisfy current debt will be matched against their respective open-items. The payment segment used to book the overpayment (on the overpayment SA) will not be matched.
- When future bills are completed, the credit balance on this "overpayment SA" will be transferred to the "real SAs" when future bills are completed (if you have plugged in the appropriate bill completion algorithm on the overpayment SA's SA type). If the overpayment satisfies all newly calculated charges, a match event is created that matches the new charges against the funds transferred from the overpayment SA. Refer to [When Are Match Events Created](#) for information about how the system creates match events at bill completion time when the new charges on the bill are satisfied by other credits (overpayments, deposit refunds, etc.).
- At some point in the future, the overpayment will be exhausted (i.e., all funds will be transferred to "real SAs"). At that point in time, the overpayment SA will close (assuming you set up the overpayment SA's SA type as a "one time"). At close time, the system creates a match event that matches the original overpayment payment segment with the adjustments that were used to transfer funds to the "real SAs". Refer to [When Are Match Events Created](#) for information about how the system creates match events when a SA closes.

Write Down Adjustment

Writing down debt is very different from [writing off debt](#). When you write down debt, you are removing the receivable with no expectation of it being paid. For example, most organizations write down small debit and credit balances as part of their write-off process (e.g., they don't send a very small amount to a collection agency).

Let's run through an example to illustrate this:

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance	SA's Payoff Balance
Bill segment created	A/R 110	0	(10)	110
	Revenue <100>			
	Tax Holding <10>			
Payment received	Cash 109.50	(9.95)	.05	0.50
	A/R <109.50>			
	Tax Holding 9.95			
	Tax Payable <9.95>			
Write down cash accounting debt	Tax Holding 0.05	(9.95)	0	0
	Write Down Expense 0.45			
	A/R <0.50>			

In order to achieve the above, you must set up an adjustment type that references a special financial transaction algorithm (refer to non-accrual accounting write down algorithms [ADJT-AD](#) and [C1-FTGL-AD](#) for more information). This

algorithm will reduce / increase the receivable balance accordingly AND cause any holding amounts to be set to zero. This adjustment type should be referenced on your write-down algorithm that is referenced on your write-off controls.

Write-Offs

At write-off time we may refund credit balances. The refunding of credit balances is handled by A/P adjustments and these have cash accounting processing as described under [Cash Refunds](#).

If we have to write-off debt, holding balances are relieved in proportion to the amount of debt that is written off (as usual). It's important to understand that for this to work, you must set up the system as follows:

- The tax holding distribution codes must have their override distribution switch turned on.
- The distribution code on the SA type associated with the service agreement to which the written-off payables are transferred must be the REAL payable distribution codes. This is important so that if the customer pays after the payables are reversed, we will be able to debit cash and credit the REAL payable distribution code.

Let's run through an example to illustrate this.

Event	Normal SA GL Accounting	Write Off Revenue SA GL Accounting	Reverse Liabilities SA GL Accounting
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>		
Write Off Time			
Reverse the held payables	Xfer 10 A/R <10>		Tax Holding 10 Xfer <10> Note, the tax holding only gets debited if you have turned on the override at write-off switch on its distribution code
Write off revenue	Xfer 100 A/R <100>	Write Off Expense 100 Xfer <100>	
If the customer subsequently pays		Cash 100 Write Off Exp <100>	Cash 10 Tax Payable <10> Note, the tax payable only gets credited if the SA type's distribution code has been defined as such

Deferred Accrual Accounting

Some implementations use a hybrid accounting method that combines cash and accrual accounting. In this case revenue, taxes, etc. are recognized on the earlier of the bill due date or the date payment is received. In this scenario, the cash accounting method is used up until the bill's due date, at which time the accrual method is enforced (let's call this "deferred accrual accounting"). A simpler flavor of deferred accrual accounting is when the revenue and liability recognition is done solely at bill due date regardless of when the payment is made.

Deferred accrual accounting affects distribution codes identified as **Use For Non-Accrual Accounting** with an associated **Accounting Method** of either Payable on Earlier of Payment or Due Date or Payable on Due Date. The system accomplishes the holding amount settlement on the bill due date using a customer class post bill completion algorithm **C1-CR-BLRVWS** that creates a bill review record to be processed on the bill's due date. The bill review batch process then analyzes these bill review records on the bill due date. If a bill review record is due for processing, the algorithm checks the outstanding balance of the holding accounts on each SA linked to the bill and creates a settlement adjustment for each SA.

Deferred Accrual Accounting Examples

The examples below illustrate the financial transactions that transpire under these different scenarios.

Example 1 - Payable On Due Date, Customer Pays In Full

The following is an example of the financial events that transpire when a customer is billed and full payment is received prior to the bill due date. The accounting method in this case is Payable On Due Date.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Payment received	Cash 110	0	(10)
	A/R <110>		
Adjustment created on bill due date	Tax Holding 10	(10)	0
	Tax Payable <10>		

Example 2 - Payable On Due Date, Customer Does Not Pay

In the following example a customer is billed and no payment is received prior to the bill due date. The bill review batch process is responsible for creating the settlement adjustment for any outstanding holding amounts on the bill's due date. The accounting method in this case is Payable On Due Date.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Adjustment created on bill due date	Tax Holding 10	(10)	0
	Tax Payable <10>		

Example 3 - Payable On Due Date, Customer Makes A Partial Payment

In the following example a customer is billed and a partial payment is received prior to the bill due date. The bill review batch process is responsible for creating the settlement adjustment for any outstanding holding amounts on the bill's due date. The accounting method in this case is Payable On Due Date.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Payment received	Cash 27.50	0	(10)
	A/R <27.50>		
Adjustment created on bill due date	Tax Holding 10	(10)	0
	Tax Payable <10>		

Example 4 - Payable On Earlier Of Payment Or Due Date, Customer Pays In Full

In the following example a customer is billed and full payment is received prior to the bill due date. The accounting method in this case is Payable On Earlier Of Payment Or Due Date.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Payment received	Cash 110	(10)	0
	A/R <110>		
	Tax Holding 10		
	Tax Payable <10>		

Example 5 - Payable On Earlier Of Payment Or Due Date, Customer Does Not Pay

In the following example a customer is billed and no payment is received prior to the bill due date. The bill review batch process is responsible for creating the settlement adjustment for any outstanding holding amounts on the bill's due date. The accounting method in this case is Payable On Earlier Of Payment Or Due Date. Note that if a payment is subsequently received after the settlement adjustment has been created, it's financial transaction(s) will not have any impact on the holding or liability accounts as these have already been booked.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Adjustment created on bill due date	Tax Holding 10	(10)	0

Example 6 - Payable On Earlier Of Payment Or Due Date, Customer Partially Pays

In the following example a customer is billed and a partial payment is received prior to the bill due date. The bill review batch process is responsible for creating the settlement adjustment for any outstanding holding amounts on the bill's due date. The accounting method in this case is Payable On Earlier Of Payment Or Due Date. Note that if a payment is subsequently received after the settlement adjustment has been created, it's financial transaction(s) will not have any impact on the holding or liability accounts as these have already been booked.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110	0	(10)
	Revenue <100>		
	Tax Holding <10>		
Payment received	Cash 27.50	(2.50)	(7.50)
	A/R <27.50>		
	Tax Holding 2.50		
	Tax Payable <2.50>		
Adjustment created on bill due date	Tax Holding 7.50	(10)	0
	Tax Payable <7.50>		

Payment Cancellations and Deferred Accrual Accounting

If a payment was responsible for transferring moneys from the holding distribution code to the payable distribution code, it stands to reason that if the payment is cancelled, it results in the reversal of this transfer from the holding distribution code to payable distribution code. If deferred accrual accounting is used and the payment is cancelled after the bill due date, the holding amounts that were transferred should remain booked.

Assume, for example, that the payment below was cancelled after the bill due date:

Event	GL Accounting
Payment received	Cash 110
	A/R <110>
	Tax Holding 10
	Tax Payable <10>

At cancellation, the above entry will be reversed, reinstating the balance in the holding account:

Event	GL Accounting
Payment cancelled after bill due date	Cash <110>
	A/R 110
	Tax Holding <10>

However, since the bill's due date has passed, the holding account needs to be booked. For open item accounts, the system comes with a customer class FT freeze algorithm ([C1-PR-CA-RVS](#)) that creates a bill review schedule for the affected bill, if one does not already exist. When the bill review batch process next runs, it checks the outstanding balance of the holding accounts on each SA linked to the bill and creates a settlement adjustment for each SA.

Event	GL Accounting
Adjustment created	Tax Holding 10
	Tax Payable <10>

Note that this solution is only applicable to open item accounting where the bill matched to the payment can be determined. If balance forward accounting is practiced, the bill or bills that the payment applied to cannot be determined. In this case, the next bill review record created for the account as part of billing will cause the balances of the holding accounts to be analyzed and the settlement will catch up at that point.

Open Item Accounting

The topics in this section provide background information about open-item accounting.

NOTE:

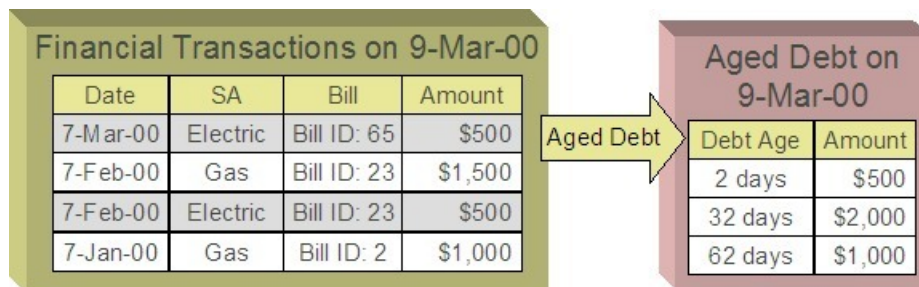
This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization practices open-item accounting. If your organization practices balance-forward accounting, you need only indicate such on your [customer classes](#) ; no other setup is required. Refer to [Open Item Versus Balance Forward Accounting](#) for more information about these two accounting practices.

Open-Item Versus Balance-Forward Accounting

If you practice open-item accounting, you match payments against bills. The term "open-item accounting" is used to describe this accounting practice because:

- Payments are matched against "open items" (i.e., unpaid bills and adjustments)
- Only unmatched bills and adjustments (i.e., open items) affect aged debt.

Contrast open-item accounting with "balance-forward" accounting - in a balance-forward world, payments are not matched to bills. Rather, payments implicitly relieve a customer's oldest debt. For example, consider the following unpaid financial transactions that exist for an account and the resultant aged debt.



In a balance-forward world, if a \$1,000 payment was made on 9-Mar-00, the customer's aged debt would look as follows:

Aged Debt on 9-Mar-00	
Debt Age	Amount
2 days	\$500
32 days	\$2,000

Notice how the \$1,000 payment relieves the 62 day old debt - it does this because, in a balance-forward world, payments payoff oldest debt first.

However, let's assume the customer wants the payment to settle his electric debt (e.g., because he disagrees with the gas bills). If you could match the \$1,000 payment to the two electric bills (i.e., open-item accounting exists), the customer's aged debt would look as follows:

Aged Debt on 9-Mar-00	
Debt Age	Amount
32 days	\$1,500
62 days	\$1,000

In sum,

- In an open-item world, payments are matched to bills and only unpaid bills and adjustments (i.e., open items) affect aged debt.
- In a balance-forward world, payments are not matched to bills and therefore a customer's aged debt is computed by aging debits (bills and adjustments) and then relieving the oldest debits using credits (payments and adjustments).

NOTE:

Financial Transactions and Bills. In an open-item world, only bill segments and adjustments are presented on a bill. When a bill is completed, only those bill segments and adjustments to be presented are swept onto a bill. Payment and payment cancellation FTs, bill segment FTs canceled before bill completion together with their corresponding bill segment cancellation FTs, and adjustment FTs marked as do not show on bill are not swept onto a bill. An adjustment's adjustment type and its algorithms determine if its FT will show on a bill by default.

Accounting Method Defined On Your Customer Classes

You define the type of accounting method that is practiced ([balance-forward versus open-item](#)) on your [customer classes](#). For example, residential customers can practice balance-forward accounting whereas industrial / commercial customers can practice open-item accounting.

Match Events

Match events are used to match open-items (i.e., debit and credit financial transactions) together. The topics in this section provide an overview of match events.

Match Events Match Debit FTs To Credit FTs

For open-item customers, the system matches credit financial transactions (FT's) to debit FT's under a [match event](#). The following is an example of a match event associated with two \$500 payments that satisfy the debt associated with one bill (on February 2000).

Match Event

Account: 10291011Status: **Balanced**

Credit FT's: \$1,000

Date	SA	FT Info	Amount
7-Mar-00	Electric	None	\$500
1-Mar-00	Electric	None	\$500

Debit FT's: \$1,000

Date	SA	FT Info	Amount
15-Feb-00	Electric	Bill ID: 22	\$1,000

Notice the following:

- The match event matches 2 credit FT's against a single debit FT. A match event may contain an unlimited number of FT's.
- The match event contains FT's associated with a single account. While the FT's under a match event may belong to multiple service agreements, all FT's under a match event must belong to the same account.
- The match event only contains bill segments that belong to a single bill. If you mix multiple bills under a single match event, then an individual bill balance cannot be properly determined when partial payments exist.
- The status of the match event is balanced. This is because the sum of the debits equals the sum of the credits. If debits do not equal credits, the status of the match event would be open and the various FT's would still affect the customer's aged debt. Refer to [Match Event Lifecycle](#) for more information.

WARNING:

It is strongly encouraged that you refrain from mixing multiple bills on a single match event. If you stick by the rule of "just one bill per match event" you will then be able to determine the outstanding balance of a partially paid bill (see the [bill page](#), bill summary section). However, if you mix more than one bill under a match event, a particular bill's balance may become indeterminate. Algorithm types have been provided which help to enforce this rule of "one bill per match event", please refer to [Match By Bill, Pay Oldest Bill First](#) for an example of a matching algorithm that enforces this notion.

When Are Match Events Created?

The following points describe when match events are created for open-item accounts:

NOTE:

Match events are only created for open-item accounts (i.e., those accounts with a customer class that indicates open-item accounting is practiced). Match events may not be created for balance-forward accounts.

- The system can create one or many match events when a payment is added. This match event matches the payment's credit FT's with the debit and credit FT's from bill segments and adjustments. The FT's that are linked to the match event are controlled by the payment's **match type** and **match value** (payments made by open-item customers must reference a match type and match value). Refer to [Payments And Match Events](#) for more information.
 - The system may create a match event when any type of financial transaction is cancelled. This match event groups together the original FT with its cancellation FT. Refer to [How Are Match Events Cancelled?](#) for more information.
 - The system creates a match event when a bill is completed for customers that pay automatically (i.e., direct debit customers). The match event groups together the bill's new charges against the automatic payment's payment segments.
 - The system creates a match event when a bill is completed where the new charges are offset by other financial transactions. For example,
 - Consider a bill that contains a deposit refund. If the sum of the deposit refund equals or exceeds the amount of the bill, the bill's FT's can be matched against the debit refunds FT's. Refer to [Refunding Deposits](#) for more information about deposit refunds.
 - Consider a bill whose new charges are offset by a previous overpayment. Refer to [Over Payments](#) for more information about overpayments.
-

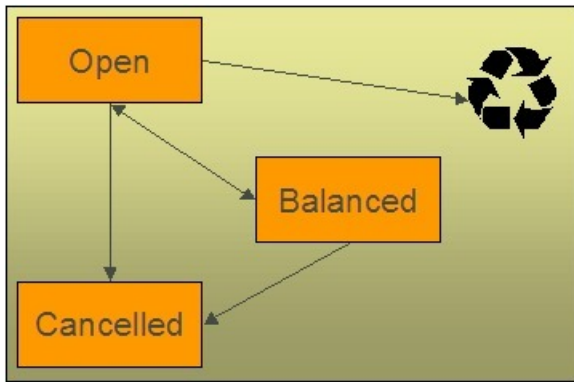
FASTPATH:

Refer to [Bill Lifecycle](#) for more information about what happens during bill completion.

- The system creates a match event when a service agreement closes and the service agreement has unmatched FT's. For example,
 - Consider a deposit service agreement that closes when the deposit is refunded to the customer. The system will create a match event with the deposit SA's FT's (the original credit and the debits used to refund the deposit) when the deposit SA closes (i.e., when its credit balance falls to zero). Refer to [Refunding Deposits](#) for more information about deposit refunds.
 - Consider a service agreement for utility debt that is written off. This service agreement closes when the system creates transfer adjustments to transfer the utility debt to a write-off service agreement (or writes down the debt). The system creates a match event to match the original debt to the transfer adjustments used to write-off the debt. Refer to [How Is Debt Financially Written Off](#) for more information about write-off processing.
- A user can create a match event manually at any time. Manual match events would be created under a variety of situations. For example:
 - If a customer disputes a charge. Refer to [Disputing Items](#) for more information about disputes.
 - To handle unusual situations when the system is unable to automatically match FT's together.

Match Event Lifecycle

The following diagram shows the possible lifecycle of a match event:



Match events are initially created in the open state. Financial transactions (FT's) linked to open match events affect arrears, but not in an open-item fashion. Rather, FT's linked to open match events affect arrears in a balance-forward fashion. Refer to [Open Item Versus Balance Forward Accounting](#) for more information about these two accounting methods.

A user may delete an open match event. When an open match event is deleted, its FT's may be linked to other match events.

The system automatically changes an open event's status to balanced when the sum of the debit financial transactions (FT's) equals the sum of the credit FT's for each SA on the match event. It's worth stressing that a match event may contain FT's from many SAs and each SA's FT's must sum to zero before the match event can become balanced.

A user may re open a balanced event (by adding / removing FT's so that the match event becomes unbalanced).

A user may cancel a balanced or open match event. Refer to [How Are Match Events Cancelled?](#) for more information about cancellation.

Payments And Match Events

As described under [When Are Match Events Created?](#), the system creates a match event when a payment is added for an open-item account. The system uses the payment's **match type** and **match value** to determine the FT's (e.g., bill segments and adjustments) that will be matched with the payment's FT's (i.e., the payment segments).

Another way to think of this is as follows:

- When most payments are distributed, the system calls the payment distribution algorithm that is plugged-in on the account's customer class.
- However, a payment that is made in respect of a specific bill requires a different distribution algorithm because the payment should only be distributed amongst the debt associated with the specific bill being paid. This is accomplished by referencing a match type / match value on the payment. The match type references the appropriate payment distribution algorithm. This algorithm is used rather than the customer class distribution algorithm.

For example, if a payment were made in respect of bill ID 192910192101, this payment would reference a match type of bill ID and a match value of 192910192101. At payment distribution time, the system calls the override payment distribution algorithm associated with this match type. The base package bill ID distribution algorithm does several things:

- It distributes the payment amongst service agreements associated with the bill.
- It creates a match event and links the bill's bill segment and adjustment FT's to it.
- Refer to the [Bill ID Match Type Algorithm](#) for more information about this algorithm.

NOTE:

The match type's distribution logic is not "hard coded". Because the match type's payment distribution logic is embedded in a plug-in algorithm, you can introduce new algorithms as per your company's requirements.

It's worth noting that payment *distribution* and *freezing* are two separate steps that typically happen in quick succession. The system's standard match event algorithms create the match event during payment distribution. This match event exists in the open state (because the payment segment's FT's have not yet been linked to the match event and therefore debit FT's do not equal credit FT's). The open match event references the debit FT's (the bill segments and adjustments) for which it pays. It is only at payment freeze time that the credit FT's (the payment segments) are linked to the match event thus allowing the match event to become balanced.

If, at freeze time, the payment's credit FT's do not equal the debit FT's on the match event, the match event is left in the open state. An alert will appear on Control Central to highlight the existence of open match events (if the appropriate alert algorithm is plugged in the installation record). In addition, you can also set up a To Do entry to highlight the existence of open match events.

Payments Are Matched To Debit Credit FTs

While the above discussion dealt with the typical situation where the payment's credit FT's are matched against a bill's debit FT's, we want to note that a payment's FT's may be matched against debit and credit FT's. Consider the following example:

Match Event			
Account: 10291011		Status: Balanced	
Bill 1929: \$2,900			
Date	SA	FT Info	Cur Amount
7-Mar-00	Electric	Bill seg	\$1,500
6-Mar-00	Gas	Bill seg	\$1,500
1-Mar-00	Gas	Adj-Credit	\$-100
Pay: \$2,900		Match Type: Bill ID 1929	
Date	SA	FT Info	Cur Amount
15-Mar-00	Electric	Pay seg	\$-1,500
15-Mar-00	Gas	Pay seg	\$-1,400

Notice that:

- The \$2,900 payment is distributed amongst two service agreements (electricity and gas).
- The FT's to which the payment segments are matched are both debit and credit FT's. Notice that the debit FT's (the bill segments) and the credit FT (the adjustment) sum to \$2,900.

Credits may result in a situation where the total amount on a bill for an SA is negative. This would be the case if in the above example the credit adjustment were for \$-1600 resulting in the total amount for the Gas SA on this bill to be \$-100 (credit). Assume a full payment of \$1400 is made towards this bill. The [Bill ID Match Type Algorithm](#) first allocates negative payment amounts to any SA credit amount on the bill being paid. It then carries over the credit amount to pay off other bill amounts. In this example, a "negative" payment segment is created to match the \$-100 credit of the Gas SA. Using the carried over credit a \$1500 payment segment is created to match the \$1500 debit of the Electric SA.

How Are Match Events Cancelled?

A user can cancel an open or balanced match event at any time. When a match event is cancelled, the event's FT's again effect arrears (and they can be associated with new match events). In other words, when a match event is cancelled, its FT's are released from the match event and become open-items.

In addition to manual cancellation, the system may automatically cancel a match event when the last of its payment FT's, if any, is cancelled (if you plug-in the appropriate FT freeze plug-in on your open-item customer classes).

For example, consider a match event that was created when a payment was made. If the payment is subsequently cancelled, the match event is also cancelled (thus releasing the match event's FT's) if no other payment FT's are linked to the match event. Please be aware that FT cancellation also causes a new match event to be created. This match event matches the original FT (the payment segment) and its cancellation FT. This means that the only "open items" that will exist after a payment is cancelled are the debit FT's that were originally paid.

NOTE:

Reopening bills associated with automatic payment customers. While many payments are cancelled due to non-sufficient funds, please be aware that if you reopen a bill for which an automatic payment was created, the system will cancel the associated payment. If this payment is associated with a match event (because the account is an open-item account), the match event will be cancelled and a new match event will be created to match the original automatic payment with its cancellation details. This is necessary because a new payment will be created with the bill is subsequently completed and this payment's FT's will be matched to the bill's FT's.

Canceling a payment can result in many match events being created. If a cancelled payment has multiple payment segments, a separate match event will be created for each payment segment.

While payment cancellation is the most common type of FT cancellation, be aware that bill segment or adjustment cancellation may also cause a new match event to be created. We don't necessarily want to always link the cancellation FT and its original FT to the same match event. For example, when the cancellation FT is swept on to the next bill it affects the next bill and not the original FT's bill. For cancellations that will not be swept on to the next bill (payment cancellation, cancellation of an adjustment that is not shown on bill, credit notes, and bill segment cancellation before the bill is completed) the system creates a new match event that matches the original FT and its cancellation FT. This way, neither FT affects aged debt. If the original FT was linked to an existing match event and no other FTs are left on this match event it is automatically canceled.

Current Amount Is Matched, Not Payoff

The system matches the current amount of financial transactions, not the payoff amount.

FASTPATH:

Please refer to [Current Amount versus Payoff Amount](#) for more information about current and payoff amounts.

Disputing Items

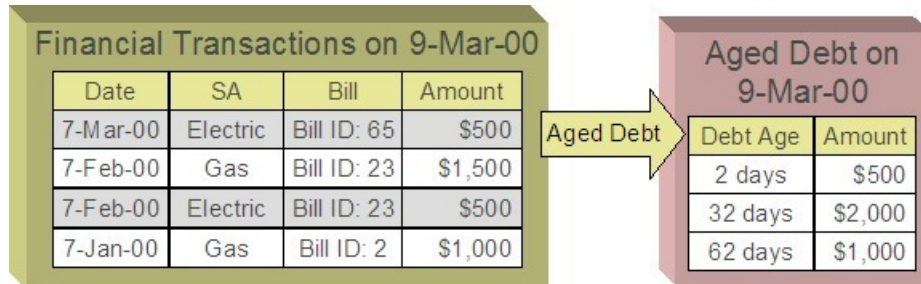
Open-item customers may dispute FT's that they are not comfortable paying. For example, a customer who receives a bill with an anomalous charge may decide to dispute it.

When an open-item customer disputes a charge, a user creates a match event and links the disputed FT(s) to it. This match event will be in the open state (because it does not contain FT's that sum to zero). In addition, the match event's "disputed switch" is turned on.

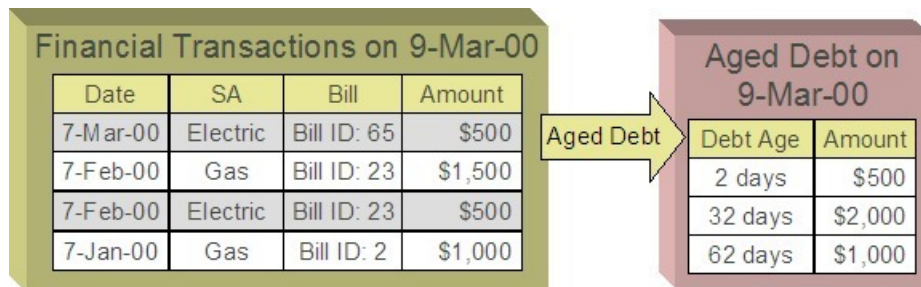
NOTE:

Alerts. An alert is displayed on control central to highlight the existence of disputed match events (if the appropriate alert algorithm is plugged in). In addition, you can also set up a To Do entry to highlight the existence of disputed match events.

While the dispute is being researched, the disputed amount will not affect aged debt, but it still forms part of the customer's balance. For example, consider the following unpaid financial transactions that exist for an account:



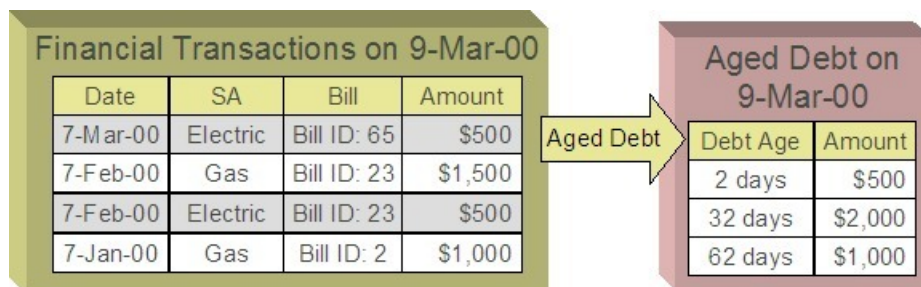
If the customer disputes the two electric bill segments, the customer's aged debt will look as follows:



Notice how a new category of debt appears - Disputed. Also notice how the 2 day old debt disappears and the 32 day old debt is reduced by the disputed amount.

The system shows disputed debt on Control Central. In addition, in all places where aged debt appears in the system, disputed debt is shown as a separate debt category.

If the dispute goes in your company's favor, the disputed match event should be cancelled (thus allowing the FT's to again impact aged debt). For example, if we assume 3 days have passed and the dispute match event is cancelled, the customer's aged debt will look as follows:



If the dispute goes in the customer's favor:

- You may decide to issue a credit note to cancel the offending bill or bill segment(s). As described above, the system in this case will automatically create new match events that match the original FTs with their cancellation FTs and cancel the disputed match event when the last item is unlinked from it.
- You may decide to cancel the offending bill segment(s) / adjustment(s). As described above, these cancellations are going to be swept on to the next bill. The system therefore will not automatically cancel the disputed match event.

Notice that the cancellation effect of the disputed items is carried over on to the next bill. This means that the previously disputed items still need to be paid.

NOTE:

Cancel / rebill. If you cancel / rebill an offending bill segment, both the cancel and the rebill will become open-items that will be matched when the next bill is paid.

- You may decide to issue an adjustment to counter the effect of the disputed FT's. In this situation, you would simply link the adjustment FT to the disputed FT's (thus allowing the match event to become balanced). It is important to use in this case an adjustment that does not show on bill.

Pay Plans

You create a [pay plan](#) when a customer agrees to make one or more scheduled payments to satisfy past (or future) debt. These payments cannot be matched to open items because it is unlikely that debit FT's exist that equal the amount of each scheduled payment. However, you must specify a match type on all payments made by open-item customers. Therefore, a conundrum exists - the system requires a match type on payments made by open item accounts, but payments made for pay plans cannot be matched to existing FT's. This conundrum is solved by the fact that match types do not have to specify an override payment distribution algorithm. The customer class's standard distribution algorithm is used for payments that reference such a match type.

You may wonder how these payments will eventually get matched to open items? If ALL payments associated with a pay plan occur before the next bill is paid (or if the pay plan exists to satisfy future debt), these payments will be swept onto the match event that is created when the customer pays their next bill. However, if the pay plan exists to payoff historical debt and this debt has not been entirely paid by the time of the next bill, an unmatched event will exist when the customer pays their subsequent bills (if the payment amount doesn't match the amount of new charges on the bill). Why? Because, the customer is not paying the entire amount of the bill and therefore the system will not be able to match the payment to open items. If this occurs, we recommend canceling the match events that are created when the customer pays their subsequent bills. When the customer finally pays off all outstanding debt, the system will create a single match event that will contain all payments and bill segments.

Over Payments

If a customer overpays a bill (i.e., we receive more cash than receivables), we strongly recommend you set up the system to NOT keep the excess credit on the customer's regular service agreements. Rather, we recommend you segregate the receivable onto an "excess credit" service agreement. If you do this, the system will transfer any excess credits to the regular service agreements at bill completion time. When this transfer occurs, the same accounting described under [Payments Segment Financial Transaction Algorithms Transfer Holding Amounts To Payable GL Accounts](#) occurs as shown in the following example. Note: this example assumes an excess credit of \$110 was transferred to a normal service agreement and the normal service agreement had \$10 of held payables.

FASTPATH:

Refer to [Overpayment Segmentation](#) for how to set up the system to segregate overpayments on a separate service agreement.

NOTE:

Why not keep excess credits on a customer's regular service agreement? Because the system can't differentiate between a credit that exists as a result of an overpayment and a credit that exists because of cancel/rebills, it would be impossible for the system to know if payables should be realized as a result of the reduced credit balance. However,

if you keep overpayments on an excess credit service agreement, the system knows to treat any transference of these credits as "payments" and therefore it can transfer holding balances to true payables.

Event	Normal SA GL Accounting	Excess Credit SA GL Accounting
Bill segment created	A/R 110	
	Revenue <100>	
	Tax Holding <10>	
Payment of \$300 is received	Cash 110	Cash 190
	A/R <110>	Overpay <190>
	Tax Holding 10	
	Tax Payable <10>	
Bill segment created	A/R 110	
	Revenue <100>	
	Tax Holding <10>	
Transfer excess credit amount to normal service agreement (when bill is completed).	Xfer 110	Overpay 110
	A/R <110>	Xfer <110>
Because the transfer adjustment is the equivalent of a cash relief outstanding tax holding is relieved in proportion to the amount of receivables that are reduced by the transfer	Tax Holding 10	
	Tax Payable <10>	
Net effect of the transfer	Xfer 110	Overpay 110
	A/R <110>	Xfer <110>
	Tax Holding 10	
	Tax Payable <10>	

NOTE:

Prepaid taxes - future functionality. If your organization allows customers to prepay taxes in anticipation of a future tax increase (the customers receive the lower rate if they pay in advance), we do not consider this prepayment to be an overpayment. Rather, it is a payment of future taxes that will be remitted to the taxing authorities at payment time (due to cash accounting). Please speak to your account manager for when corresponding functionality will be available.

Setting Up The System To Enable Open Item Accounting

The following section provides an overview of how to enable open-item accounting.

Match Type Setup

The number of match types that you will need is dependent on the number of ways you want payments to be matched to open items. At a minimum, you will probably need the following match types:

- **Bill ID.** This match type should reference an override payment distribution algorithm that distributes the payment based on the bill ID specified on the payment (in match value). Refer to [Payments And Match Events](#) for more information.

- **SA ID.** This match type should reference an override payment distribution algorithm that distributes the payment based on the SA ID specified on the payment (in match value). Refer to [Payments And Match Events](#) for more information.
- **Pay Plan.** This match type should NOT reference an override payment distribution algorithm (if this algorithm is blank, the customer class's payment distribution algorithm is used). Refer to [Pay Plans](#) for more information.

Match Event Cancellation Reason Setup

The number of match event cancellation reasons that you will need is dependent on the number of ways your organization can justify the cancellation of a match event. At a minimum, you will probably need the following match event cancellation reasons:

- **FT Cancellation.** This cancel reason should be referenced on the Customer Class FT Freeze algorithm that is responsible for canceling match events when one of its financial transactions is cancelled.
- **Incorrect Allocation.** This cancel reason should be specified by users when they cancel match events that were created by the system erroneously.

Customer Class Setup

The following points describe [customer class](#) oriented set up functions:

- Turn on the open-item accounting switch.
- Set up the following algorithms for each CIS division:
 - Specify a **payment freeze** algorithm that causes a payment's FT's to be linked to the match event that was created when the payment was distributed. Refer to [Payments And Match Events](#) for more information.
 - Specify a **FT freeze** algorithm that causes match events to be cancelled (and a new match event to be created) when a FT is cancelled. Refer to [How Are Match Events Cancelled](#) for more information about cancellation.
 - We strongly recommend specifying an **overpayment** algorithm that causes overpayments to be segregated onto an "excess credit / overpayment" SA. Refer to [Overpayments](#) for more information.

Overpayment SA Type Setup

Specify a **bill completion** algorithm that causes the credit amount on overpayment SAs to be transferred to newly create debt (created when the bill is created). This algorithm transfers an overpayment SA's balance to regular SAs and creates a match event if the overpayment covers the entire bill. Refer to [Overpayments](#) for more information.

Installation Record Setup

Specify an **automatic payment** algorithm that causes a match event to be created when automatic payments are created for open-item accounts. The base package algorithm will do this for you if you specify the appropriate parameter on the algorithm. Refer to [APAY-CREATE](#) for more information about this algorithm.

If you want a Control Central alert to highlight when the current account has any open match events, plug in the appropriate **control central alert** algorithm on your installation record. Refer to [C1-OPN-MEVT](#) for more information about this algorithm.

If you want to enable manual pay segment distribution for open item accounts, along with other functions, you will need to plug in an installation algorithm for bill balance calculation. Refer to [C1-OI-BI-AMT](#) for more information about this algorithm.

To Do Entry Setup

Two To Do types are supplied with the base package:

- **TD-MODTL**. This To Do type highlights the presence of open, disputed match events.
- **TD-MONTL**. This To Do type highlights the presence of open, non-disputed match events.

Each of the above To Do types should be configured with the roles that work on entries of each type.

In addition, the account management group and/or divisions from which the default roles are extracted should be updated to define the role that should be defaulted for each of the above To Do types.

FASTPATH:

Refer to [The Big Picture Of To Do Lists](#) for more information about To Do lists.

Setting Up Match Types

Most payments are distributed amongst service agreements using the payment distribution algorithm specified on the payment's account's customer class. This algorithm decides how to distribute a payment amongst an account's existing debt if the customer doesn't specify how the payment should be distributed.

A customer can specify how a payment is distributed by specifying a match type and match value on their payments. Consider the following examples:

- Customers that are subject to open-item accounting (this is defined on the account's customer class) tell the system exactly which debt is covered by their payments. For example, an open-item customer might make a payment in respect of bill ID 123919101919.
- Even non open-item customers can direct payments to specific SAs. For example, the system allows a balance-forward customer's payment to be directed to a specific service agreement (however, they cannot direct payments to specific bills as only open-item customers can do this).

Match types are used to define the specific type of debt that is covered by a payment. The match type contains the algorithm that effectively overrides the standard payment distribution algorithm defined on the account's customer class.

NOTE:

Background information. Please refer to [Payments And Match Events](#) and [Match Type Setup](#) for more information about how match types are used.

To set up match types, select **Admin > Financial > Match Type**.

Description of Page

Enter an easily recognizable **Match Type** and **Description**.

Define the **Pay Dist Override Algorithm** used to distribute payments that reference this match type. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that overrides the normal payment distribution algorithm.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MATCH_TYPE](#).

Setting Up Match Event Cancellation Reasons

When a match event is canceled, a cancel reason must be supplied.

NOTE:

Background information. Refer to [How Are Match Events Cancelled?](#) and [Setting Up Match Event Cancellation](#) for more information about cancellation.

To set up match event cancellation reasons, select **Admin > Financial > Match Event Cancel Reason**.

Description of Page

Enter an easily recognizable **Match Event Cancel Reason** and **Description**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MEVT_CAN_RSN](#).

Fund Accounting

The topics in this section provide background information about fund accounting.

NOTE:

This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization practices fund accounting (this type of accounting is typically performed by municipal utilities). If your organization does not practice fund accounting, you need only indicate such on the [Installation Record](#) ; no other setup is required.

Fund Accounting Overview

Municipal utilities, and not-for-profit organizations in general, often use a form of accounting different from that used by for-profit corporations. Municipal utilities typically practice fund accounting, whereas corporations practice corporate accounting.

Regulations or other restrictions may require a municipal utility to account for the finances of each of its departments as a separate entity. If a municipal utility provides both water and wastewater service, a municipal utility may need to track the receivables, revenue, and liabilities for water service separately from those of wastewater. In contrast, a corporation is free to co-mingle the moneys of the two services.

To track the services separately, the municipal utility sets up a fund for each department. A fund is an accounting entity with its own self-balancing set of accounts. Each fund has its own "sub general ledger" with its own chart of accounts, and within each fund, its debits equal its credits at all times. This allows the utility to report on the financial state of each fund independently.

In addition to having a fund for each department, there is also a general fund, which is used to handle inter-fund transfers as well as shared accounts.

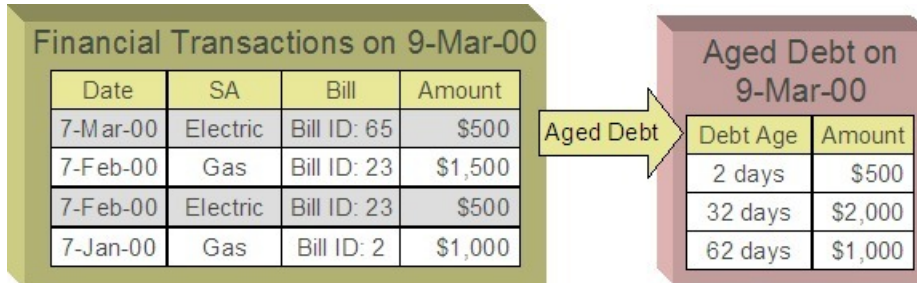
Fund Accounting Example

Consider a municipal utility which provides water and wastewater service. The utility has two departments: water and wastewater. Each department must track their finances separately therefore a fund is setup for each department:

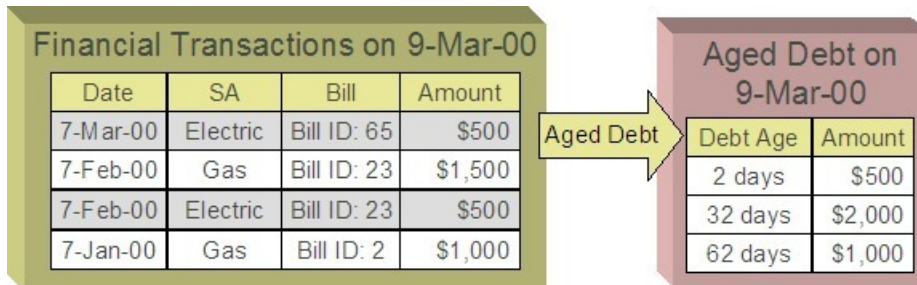
- Water (fund 01).
- Wastewater (fund 02).

In addition, with fund accounting, there is always a general fund (fund 99).

Assume the following bill is generated.



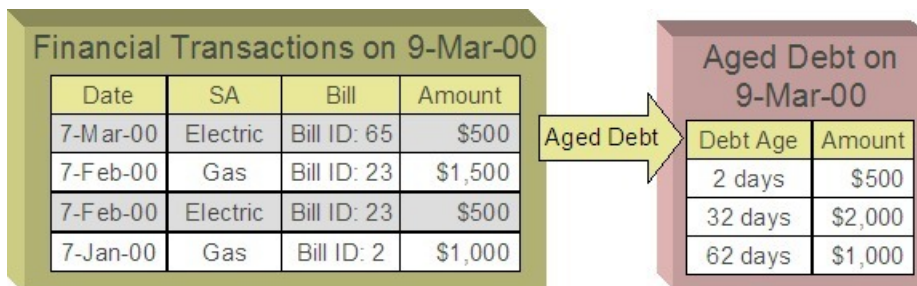
The bill would produce the following GL entries:



For each fund, the GL details of the bill will include a debit to the accounts receivable (A/R) account and credits to the revenue and taxes payable accounts. In organizational terms, each department is owed a portion of the overall bill by the customer, part of which is sales by the department and part of which is owed to the taxing authorities by the department. Each fund is balanced.

Note that the accounting could be identical under corporate accounting if each service is its own division with its own chart of accounts.

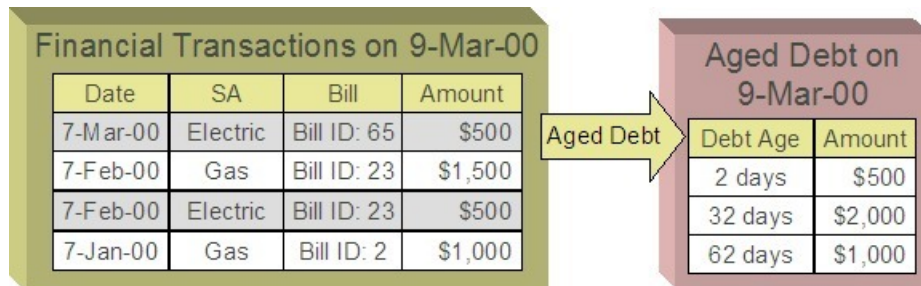
The following diagram illustrates the initial GL accounting that would occur when the payment arrives:



The utility's general cash account is debited, and the departmental funds' A/R accounts are credited. In other words, the cash is held by the utility as a whole but the receivables are reduced for the individual departments.

If the accounting were left in this state, the fund accounting principal - that each fund represents an independent entity with a self-balancing chart of accounts - would be violated. This violation is caused due to the fact that cash is recorded on the general fund, not the departmental funds, causing the general fund to have an excess debit and the departmental funds to have an excess credit.

From an organizational viewpoint, to make each department whole, the departments need to note what portion of the cash they own, and correspondingly, the utility needs to note what portion of the cash is owed to each department. The following diagram illustrates this point.



To maintain a balance of debits and credits within each fund, the departmental funds have an "equity in pooled cash" (EPC) account and the general fund has a liability account for each departmental fund. In addition to debiting the general fund's cash account and crediting the departmental funds' A/R accounts, the departmental funds' EPC accounts are debited and the general funds liability accounts are credited.

And so, with the additional GL entries, all funds have matching debits and credits.

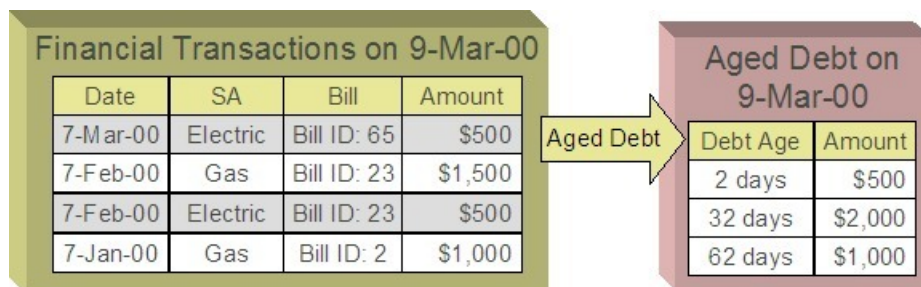
An Example Of A Bill Segment That References Multiple Funds

Consider a municipal utility that primarily supplies water service but is also responsible for maintaining the city's fire hydrants. The costs for fire hydrant maintenance are borne by the water customers and make up just a small portion of the overall bill. These costs are simply added to the water bill as a line item. The utility has two departments:

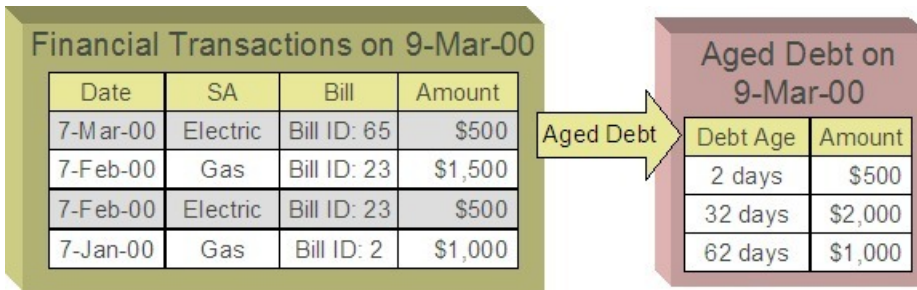
- Water service (fund 01)
- Hydrant maintenance (fund 39).

In addition, there is a general fund (fund 99).

Assume the following bill is generated for water and hydrant services.



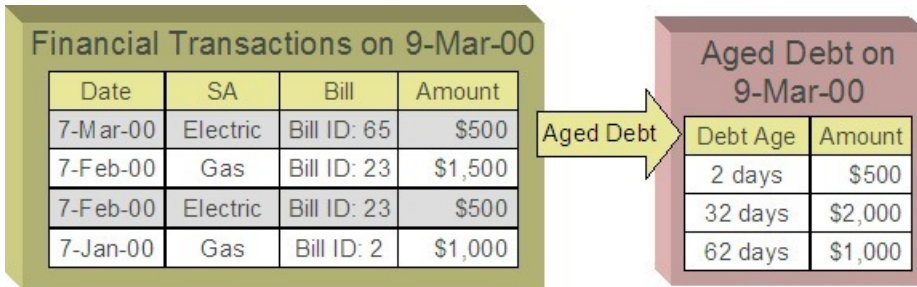
The following diagram illustrates the initial GL entries for the bill:



In accounting for the bill, the water fund's A/R is debited, the water and hydrant funds' revenue accounts are credited, and the water's taxes payable account is credited.

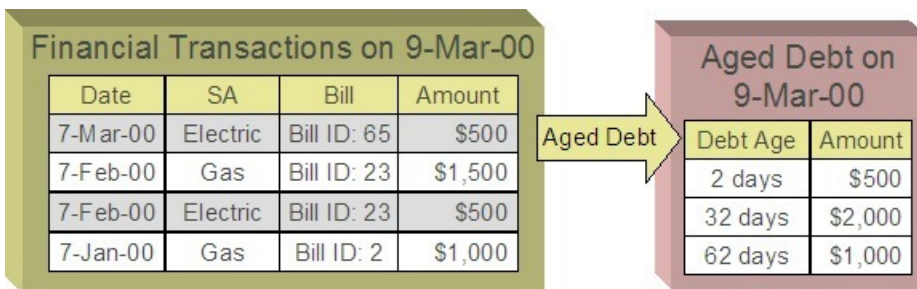
If left at this, the funds would be out of balance; the water fund would have an overall excess debit and the hydrant fund would have an equal excess credit. In organizational terms, the hydrant fund has recorded sales but that amount is recorded as being owed to the water department.

To balance each department, the water department accepts the responsibility for collecting the hydrant charges from the customer but immediately remunerates the charges to the hydrant fund.

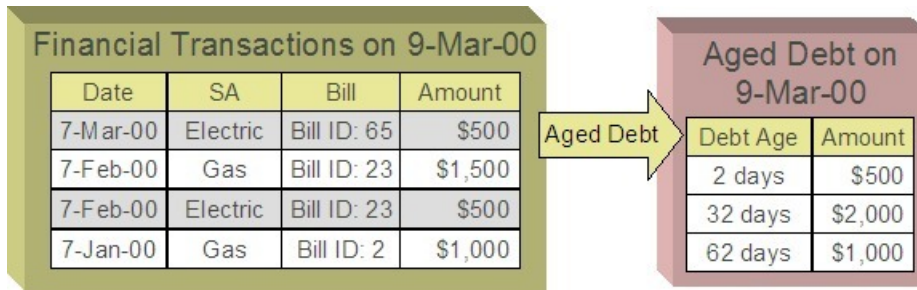


This transfer is done using the general fund. The water fund's EPC account is credited and the liability to water is debited with the amount of the hydrant revenue. Also, the hydrant fund's EPC account is debited and the general fund's liability to hydrant account is credited by the hydrant revenue. In effect, the water department owes the hydrant charges to the general fund, and the general fund owes the hydrant charges to the hydrant fund.

The following diagram illustrates the initial GL accounting that would occur when the payment arrives:



When the payment arrives, the cash is debited to the general fund's cash account, and the water fund's A/R is relieved. Again, the funds would be unbalanced if left in this condition; the water fund would have an excess of credits and the general fund would have an excess of debits.



To maintain each fund's balance of debits and credits, the general fund's liability to the water fund is credited by the amount of the department's share of the cash, and the water fund's EPC is debited. Note that the payment has no effect on hydrant fund's EPC and the general fund's liability to the hydrant fund. The hydrant department "received" its money from the water department when the bill was created.

And so, all funds have matching debits and credits.

Accounting Method Is Defined On The Installation Options

You must turn on a switch on the [Installation Record](#) to enable fund accounting.

Fund Controls Fund-Balancing Entries

There are two levels of debit and credit balancing in fund accounting. There is the balancing required by double entry accounting: the total debits in the entire GL must equal the total credits. This is required regardless of whether fund or corporate accounting is used. The distribution codes for these entries come from varying sources, depending on the type of financial event.

FASTPATH:

Refer to [The Source Of GL Accounts On Financial Transactions](#) for information on the sources of the distribution codes.

The second level of balancing is specific to fund accounting. Within each fund-not just across the GL-the total debits must equal the total credits. The original distribution code from the financial event has a fund specified. For example, a bill would cause a debit to a fund's A/R distribution code, and included in that A/R distribution code is the fund. It is the definition of the fund that specifies whether fund-balancing entries are required and provides the distribution codes for these entries.

For a departmental fund, the fund-balancing debit and credit would be specified. When a debit is applied to a departmental fund's GL account, an additional account (typically the general fund's liability to the departmental fund) is debited and an account (typically the departmental fund's EPC) is credited. When a credit is applied to a departmental fund's account, an additional account (typically the general fund's liability to the departmental fund) is credited and an account (typically the department's EPC) is debited.

For the general fund, no fund-balancing debits and credits are specified.

Building Fund-Balancing GL Details

Building the GL details for a financial event is a two-step process.

- First, the system generates the regular GL details for a financial transaction (FT). This is done regardless of whether corporate or fund accounting is used.

- Second, if fund accounting is activated (by turning on a switch on the [Installation Record](#)), the system analyzes the distribution code on each GL detail associated with the FT. If a [fund](#) is specified on a distribution code, the system checks the definition of the fund. If fund-balancing entries are specified on the fund, two additional GL entries are added to the FT:
- An offsetting entry to the Equity in Pooled Cash account is created for the departmental fund (e.g., if the FT is debiting a given fund, an offsetting credit is created in the funds EPC account).
- Another entry to the departments Liability account is created for the general fund.

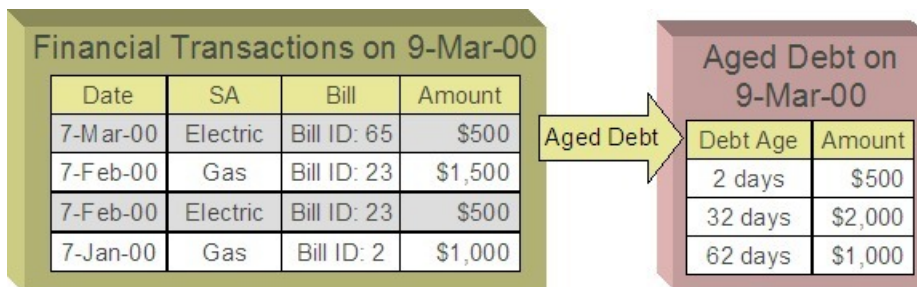
The result is a consolidated set of GL entries for the FT, incorporating the regular entries as well as the fund-balancing entries.

The topics in this section illustrate the generation of the GL details for the earlier examples.

FTs Whose GL Details All Reference The Same Fund Do Not Impact the General Fund or EPC Accounts

In [Fund Accounting Example](#), where the bill's bill segments reference a single fund, the system creates a fund-balancing GL entry for each GL entry applied to a departmental fund:

- A debit to a departmental GL account triggers a debit to the general fund's liability-to-departmental-fund account and a credit to the departmental fund's equity-in-pending-cash account.
- A credit to a departmental GL account triggers a credit to the general fund's liability-to-departmental-fund account and a debit to the departmental fund's equity-in-cash account.

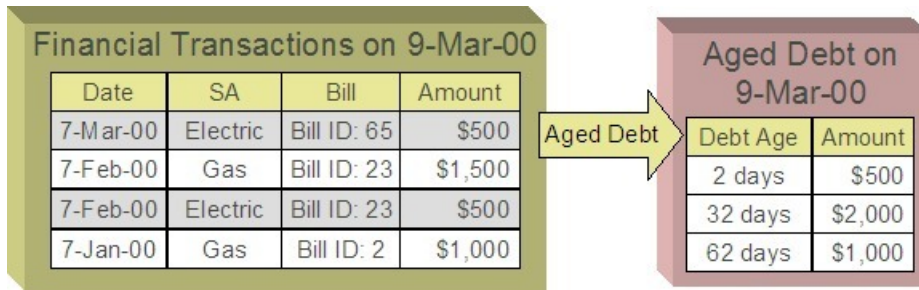


The net effect on the individual equity-in-cash and general fund's liability accounts is zero because the debits and credits net to zero for each GL account. In other words, the yellow boxes net to zero and therefore fund accounting does not impact the bill segment's financial transactions. Refer to [Fund Accounting Example](#) for the resulting consolidated GL entries.

An FT Whose GL Details Reference Multiple Funds

In [An Example Of A Bill Segment That References Multiple Funds](#), where the bill's bill segments reference multiple funds (water and hydrant), the system also creates fund-balancing GL entries for the financial transaction:

- A debit to a departmental GL account triggers a debit to the general fund's liability-to-departmental-fund account and a credit to the departmental fund's equity-in-pending-cash account.
- A credit to a departmental GL account triggers a credit to the general fund's liability-to-departmental-fund account and a debit to the departmental fund's equity-in-cash account.



The net effect of the bill on the GL is that the water fund's EPC has a credit of \$0.80, the hydrant fund's EPC has a debit of \$0.80, the general fund's liability to the water fund has a debit of \$0.80, and the general fund's liability to the hydrant fund has a credit of \$0.80. Note that, overall, the general fund's overall liability to the departmental funds nets to zero. Refer to [An Example Of A Bill Segment That References Multiple Funds](#) for the resulting consolidated GL entries.

Setting Up The System To Enable Fund Accounting

The following section provides an overview of how to enable fund accounting.

Turn On Fund Accounting

On the [Installation Record](#), indicate that fund accounting is Practiced.

Defining Funds

A fund must be setup for each specific fund in your organization. Don't forget to also set up a fund for the general fund. Navigate using **Admin > Financial > Fund**.

Description of Page

Enter a **Fund** and a **Description** to identify the fund.

If this fund is used to balance other funds or to hold cash, indicate a **Fund Type** of General, otherwise indicate that it is Specific.

If the fund type is Specific, specify the **Equity Distribution Code** and **Liability Distribution Code**. These codes are used to balance financial transactions that span funds. The equity distribution code should belong to the same **Fund** as the one you are defining. The liability distribution code should belong to the general fund.

Distribution Codes Must Include Fund ID

All of your distribution codes must include their respective fund ID.

FASTPATH:

For more information, refer to [Setting Up Distribution Codes](#).

Update Your Funds With Their Respective Equity and Liability Distribution Codes

After distribution codes have been setup, you must update your funds to indicate the equity and liability accounts used to balance inter-fund financial transactions.

United Kingdom VAT and CCL

The topics in this section provide information about value added tax (VAT) and climate change levy (CCL) charges that are specific to non-domestic customers in the United Kingdom market.

NOTE:

Applicable for UK market. This section is only relevant for the United Kingdom market. Other markets may disregard this section.

UK VAT Overview

Two rates of VAT, referred to as standard rate VAT and reduced rate VAT, are applicable to energy related charges in the UK. Domestic (i.e., residential) customers always pay VAT at the reduced rate. Non-domestic customers on the other hand normally pay standard rate VAT. However, part or all of the energy related charges for a non-domestic customer might be subject to reduced rate VAT. De minimis and VAT declarations affect the percentage of the energy related charges that is subject to each VAT rate:

- If average daily usage at a premise for a given service type does not exceed a certain threshold (the de minimis limit), all energy related charges at that premise / service type are taxed at the reduced rate.
- Some non-domestic customers, such as those with a mixed-use premise, may be eligible to pay reduced rate VAT on part or all of their energy related charges. Customers may file a VAT declaration specifying the percentage of their energy related charges that are eligible for reduced rate VAT - refer to [Maintaining Declarations](#) for more information. If the declared percentage exceeds a given threshold, the declared percentage is deemed to be 100% and the customer pays VAT at the reduced rate on all energy related charges at that premise. VAT declarations are non-transferable and must be filed for each account, premise, and service type combination.

In addition to the potential for different rates of VAT to be applicable on a bill, UK tax regulations require that excess credits are considered a prepayment of energy related charges together with VAT. Refer to [Excess Credits and UK VAT](#) for more information.

The system comes supplied with various algorithms types that can be used to perform the VAT calculations.

UK CCL Overview

The climate change levy (CCL) is based on the amount of energy used that is subject to standard rate VAT. Similarly to VAT declarations, customers may file for exemption from CCL. A CCL declaration specifies the percentage of the CCL charges that the customer is exempt from - refer to [Maintaining Declarations](#) for more information. CCL declarations are non-transferable and must be filed for each account, premise, and service type combination.

CCL charges themselves are subject to standard rate VAT.

UK VAT and CCL Bill Examples

The following examples show how a bill for a non-domestic customer is affected by de minimis, VAT declarations, and CCL declarations. In the examples, standard rate VAT of 17.5%, reduced rate VAT of 5%, a VAT declaration threshold of 60%, and a CCL charge of 0.43p per unit of energy are used.

Example 1 - Normal Account

This example shows the bill for a normal non-domestic account with no declarations and consumption above the de minimis limit. Standard rate VAT is applied to all energy related charges (the standing charge and the per unit charge) and to the CCL charge.

Bill Line	Amount
Standing Charge	10.00
2,000 units @ 10p	200.00
CCL on 2,000 units @ 0.43p	8.60
VAT @ 17.5% on 218.60	38.25
Total	256.85

Example 2 - Account with Consumption Under the De Minimis Limit

In this example, consumption does not exceed the de minimis limit and therefore reduced rate VAT is applied to all the energy related charges. There is no climate change levy because only units subject to standard rate VAT are subject to CCL.

Bill Line	Amount
Standing Charge	10.00
2,000 units @ 10p	200.00
VAT @ 5% on £210.00	10.05
Total	220.05

Example 3 - Account with VAT Declaration

This example is for a non-domestic account with a VAT declaration of 20%. Consumption is above the de minimis limit. CCL applies to only 80% of the total units as 20% is subject to reduced rate VAT and therefore exempt. 80% of the energy related charges (the standing charge and the per unit charge) and all of the CCL charge are subject to standard rate VAT. 20% of the energy related charges are subject to reduced rate VAT.

Bill Line	Amount
Standing Charge	10.00
2,000 units @ 10p	200.00

CCL on 1,600 units @ 0.43p	6.88
VAT @ 17.5% on £174.88	30.60
VAT @ 5% on £42.00	2.10
Total	249.58

Example 4 - Account with VAT Declaration and CCL Declaration

This example is for a non-domestic account with a VAT declaration of 20% and a CCL declaration of 10%. Consumption is above the de minimis limit. CCL applies to only 80% of the total units as 20% is subject to reduced rate VAT and therefore exempt. The customer gets a credit (CCL relief) for 10% of the CCL charges as a result of the CCL declaration. 80% of the energy related charges (the standing charge and the per unit charge) and all of the CCL charge less the CCL reliefs are subject to standard rate VAT. 20% of the energy related charges are subject to reduced rate VAT.

Bill Line	Amount
Standing Charge	10.00
2,000 units @ 10p	200.00
CCL on 1,600 units @ 0.43p	6.88
CCL relief	<0.68>
VAT @ 17.5% on £174.20	30.48
VAT @ 5% on £42.00	2.10
Total	248.78

Billing and UK VAT

The following sections describe how VAT rules are implemented when a bill is produced.

Application of De Minimis

The de minimis rule specifies that all energy related charges at a premise be subject to reduced rate VAT if the total usage at a premise does not exceed a certain threshold. To determine if de minimis applies, the total billed consumption for a premise must therefore be known. However, energy usage at a premise may be measured with several meters and billed by multiple service agreements. To ensure that the final billed consumption for the premise is used to determine if de minimis applies, all service agreements for an account of a given service type at a given premise should be billed together and de minimis should be checked after the bill segments have been generated. In addition, all bill segments on one bill for consumption at a given premise must be for the same bill period.

NOTE:

Bill segments from different periods may not appear on the same bill. If a bill segment is canceled, all bill segments associated with consumption at a given premise for the same service type must be canceled and re-billed together. You cannot re-bill the bill segment on a bill for a different period.

An algorithm type is supplied with the base package that checks for de minimis at bill completion time. The base bill completion algorithm type checks that for any service agreement that was billed, bill segments for all the account's service agreements with the same service type and characteristic premise exist on the bill. It then calculates the total consumption for the premise and determines if de minimis applies. If de minimis applies, it sets a bill characteristic whose value is the premise ID to indicate that de minimis applies and regenerates the bill segment. Refer to the algorithm type [CPBC-DMCH](#) for more information on how this type of algorithm operates.

The calculation rule calculation algorithm type that calculates standard and reduced rate VAT ([RCAM-VAT](#)) applies the de minimis rule if the bill characteristic for de minimis is found. Refer to [Calculation of VAT](#) for more information. Note that during the initial generation of each bill segment, the characteristic will not exist and standard rate VAT will be applied. This means that if you look at a bill before it is complete, VAT may not be accurately reflected.

NOTE:

Batch billing cannot regenerate re-billed bill segments. If you cancel and re-bill a bill segment, you should complete the bill on-line so that the re-billed bill segments can be regenerated if de minimis applies.

Calculation of VAT

To calculate VAT, the percentages of energy related charges subject to standard rate VAT and reduced rate VAT must be determined. As these percentages vary from customer to customer and even from one bill to the next for the same customer, their calculation must take place at billing time and is handled by a calculation rule calculation algorithm.

A calculation rule calculation algorithm type [RCAM-VAT](#) is provided to calculate the percentages of energy related charges subject to standard rate and reduced rate VAT, taking into account de minimis and any VAT declaration that is in effect for the service type, account, and premise. The percentages can then be applied to the appropriate charges that are cross-referenced.

Calculation of CCL

CCL is a charge per unit of energy subject to standard rate VAT. It is therefore dependent on the same percentage of energy related charges subject to standard rate VAT determined during the calculation of VAT. A calculation rule calculation algorithm handles calculation of CCL.

A calculation rule calculation algorithm type [RCAM-CCL](#) is provided to calculate CCL charges and CCL relief, taking into account any CCL declarations that are in effect for the service type, account, and premise.

Excess Credits and UK VAT

When a financial transaction that results in a credit balance for a service agreement is frozen, the amount of the excess credit must be accounted for as a prepayment of energy related charges and VAT. VAT is calculated at the reduced rate for domestic customers and at the standard rate for non-domestic customers. When the excess credit is used, the VAT liability is reversed.

A customer class FT freeze algorithm type [CFTZ-VAT-GL](#) is provided to create additional GL detail entries for unbilled revenue and VAT liability when the freezing of an FT results in a credit balance or a change to a service agreement's credit balance.

Excess Credit GL Accounting Example

The following example shows the additional GL details that are created when a service agreement's balance changes and the starting or ending balance is a credit. For this example, a single VAT rate of 10% is used simply to illustrate the principle.

The service agreement has a zero starting balance. Note the following:

- When the first payment is received, the overpayment results in a credit balance and additional GL entries are created to recognize the unbilled energy revenue and unbilled VAT liability.
- When the second bill segment is created, the credit balance is reduced and additional GL entries are created to back out the unbilled GL entries, up to the amount of the credit balance.
- When the bill segment is canceled, the service agreement again has a credit balance and additional GL entries are created to recognize the unbilled energy revenue and unbilled VAT liability.

Event	Normal GL Accounting	Additional GL Accounting	SA Balance
Bill segment for £110 created	A/R 110 Revenue <100> VAT <10>		110
Payment of £330 is received	Cash 330 A/R <330>	A/R 220 Unbilled Energy <200> Unbilled VAT <20>	<220>
Bill segment for £275 created	A/R 275 Revenue <250> VAT <25>	A/R <220> Unbilled Energy 200 Unbilled VAT 20	55
Payment of £55 is received	Cash 55 A/R <55>		0
Bill segment for £275 is canceled	A/R <275> Revenue 250 VAT 25	A/R 275 Unbilled Energy <250> Unbilled VAT <25>	<275>

Setting Up The System For UK VAT and CCL

The following sections provide an overview of how to configure the system to apply UK VAT and CCL charges for non-domestic customers.

Bill Segment Freeze Installation Option

Select Freeze at Bill Completion as the bill segment freeze option. The system determines if de minimis applies at bill completion time and has to regenerate the bill segments as necessary.

Customer Class

Create a separate customer classes for non-domestic customers.

Service Agreements

You must configure the system so that each service agreement that must be checked for de minimis is only linked to service points at a single premise. The service agreement must reference that premise as its characteristic premise. Refer to [Application of De Minimis](#) for more information.

Bill Factors for UK VAT and CCL

- Standard Rate VAT
- Reduced Rate VAT
- VAT Declaration Percentage Threshold. Set up a separate bill factor for each service type as the thresholds differ based on service type.
- De Minimis Average Daily Amount Threshold. Set up a separate bill factor for each service type as the de minimis amounts differ based on service type.
- Climate Change Levy (CCL) Per Unit Price. Set up a separate bill factor for each service type as the CCL per unit prices differ based on service type.

Characteristic Type for De Minimis Amounts

Set up a characteristic type to use for the bill characteristic that indicates when de minimis applies. Define the type of characteristic value as a Foreign Key Value and specify a foreign key reference that points to the premise table. Specify this characteristic type as parameters to the algorithms that apply the de minimis rule and that apply VAT.

You need to define a characteristic type for every service type that is subject to the de minimis rule, as the de minimis limit may apply for one service type but not another on the same bill.

Distribution Code

Define the following distribution codes:

- Unbilled Prepaid Energy Related Charges
- Unbilled Prepaid VAT

These distribution codes are required as parameters to the algorithms to create the GL details for excess credit.

UOM / TOU / SQI

The algorithm that calculates CCL requires the following service quantities:

- The percentage of energy related charges that is subject to standard rate VAT
- Define the SQI used to store this service quantity. You will specify this SQI as a parameter to the algorithm used to calculate VAT. Refer to the [RCAM-VAT](#) algorithm type for more information about the base package algorithm.
- The total consumption
- Define the UOM / TOU / SQI used to store these service quantities.

Service Quantities

The algorithms that calculate VAT and CCL require a service quantity containing the total consumption. Unless you have registers that measure the total consumption independent of time of use or interval, you will need to set up a pre-processing calculation rule or rate calculation algorithm to aggregate the usage into one service quantity.

Algorithms for UK VAT and CCL

Add the following [algorithms](#):

- Apply De Minimis Rule. Define an algorithm for each service type for which de minimis should be checked. Plug the algorithm(s) in on the customer class for the Pre Bill Completion system event. Refer to the [CPBC-DMCH](#) algorithm type for more information about the base package algorithm.
- Apply VAT to Cross-Referenced Calculation Rules. Define one algorithm to calculate VAT at the standard rate and one algorithm to calculate VAT at the reduced rate (specify whether to calculate standard rate or reduced rate VAT using the algorithm parameter). You will need to define a set of algorithms for each service type (the bill characteristic to store the premise when de minimis applies and the VAT Declaration percentage threshold bill factor referenced in the algorithm parameters are different for each service type). Plug these algorithms in on the calculation rule for the Calculation Algorithm system event. Refer to the [RCAM-VAT](#) algorithm type for more information about the base package algorithm.
- Create Excess Credit GL Details. Plug this algorithm in on the customer class for the FT Freeze system event. Refer to the [CFTZ-VAT-GL](#) algorithm type for more information about the base package algorithm.
- Calculate CCL. Plug this algorithm in on the calculation rule for the Calculation Algorithm system event. Refer to the [RCAM-CCL](#) algorithm type for more information about the base package algorithm.
- Highlight Effective Declarations for Account and Premise. Plug this algorithm in on the installation option for the Control Central Alert system event. Refer to the [CCAL-DECL](#) algorithm type for more information about the base package algorithm.

Calculation Rules to Charge VAT and CCL

Four additional calculation rules are required to charge for VAT and CCL on non-domestic rates:

- Calculate standard rate VAT
 - Calculation rule type = Calculation Algorithm
 - Value type = Percentage

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for standard rate VAT.

- Specify the UOM/TOU/SQI of the service quantity that holds the total consumption.
- Turn on Derive SQ
- Calculation algorithm = the algorithm you set up to calculate VAT at the standard rate
- Cross-reference all calculation rules that contribute to the total bill amount for energy related charges
- Note that the algorithm overrides description on bill
- Calculate reduced rate VAT
 - Calculation rule type = Calculation Algorithm

- Value type = Percentage

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for reduced rate VAT.

- Specify the UOM/TOU/SQI of the service quantity that holds the total consumption.
- Turn on Derive SQ
- Calculation algorithm = the algorithm you set up to calculate VAT at the reduced rate
- Cross-reference all calculation rules that contribute to the total bill amount for energy related charges
- Note that the algorithm overrides description on bill
- Calculate CCL and CCL Relief
 - Calculation rule type = Calculation Algorithm
 - Value type = Unit Rate

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for CCL per unit price.

- UOM/TOU/SQI = the identifier of the service quantity containing the total units of energy consumed
- Calculation algorithm = the algorithm you set up to calculate CCL
- Note that the algorithm overrides description on bill
- VAT on CCL
 - Calculation rule type = Apply To
 - Value type = Percentage

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for standard rate VAT.

- Cross-reference the calculation rule that calculates CCL and CCL Relief

Setting up the calculation rules as above produces separate lines for VAT on energy related charges and on CCL (i.e., lines will be created for standard rate VAT on energy related charges, reduced rate VAT on energy related charges, and standard rate VAT on CCL charges).

You can set up rates to calculate standard rate VAT on energy related charges and CCL together as follows:

- Calculate standard rate VAT charges
 - Calculation rule type = Calculation Algorithm
 - Turn on For Calculation Purposes Only (Result Type = Charge)
 - Value type = Percentage

Define a value source of Value and a value of 100.

- Specify the UOM/TOU/SQI of the service quantity that holds the total consumption.
- Turn on Derive SQ
- Calculation algorithm = the algorithm you set up to calculate VAT at the standard rate
- Cross-reference all calculation rule that contribute to the total bill amount for energy related charges.
- Calculate CCL and CCL Relief
 - Calculation rule type = Calculation Algorithm
 - Value type = Unit Rate

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for CCL per unit price.

- UOM/TOU/SQI = the identifier of the service quantity containing the total units of energy consumed
- Calculation algorithm = the algorithm you set up to calculate CCL

- Note that the algorithm overrides description on bill.
- Calculate standard rate VAT
 - Calculation rule type = Apply To
 - Value type = Percentage

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for standard rate VAT.

- Cross-reference the calculation rule that calculates standard rate VAT charges and the calculation rule that calculates CCL and CCL Relief
- Calculate reduced rate VAT:
 - Calculation rule type = Calculation Algorithm
 - Value type = Percentage

It is recommended that you use a value source of Bill Factor and reference the bill factor you set up for reduced rate VAT.

- Specify the UOM/TOU/SQI of the service quantity that holds the total consumption.
- Turn on Derive SQ
- Calculation algorithm = the algorithm you set up to calculate VAT at the reduced rate
- Cross-reference all calculation rules that contribute to the total bill amount for energy related charges.
- Note that the algorithm overrides description on bill.

Bill Taxation Threshold

Some implementations only apply taxes if the accumulated tax amount at the bill level exceeds some specified threshold amount.

Taxation Threshold Examples

The following examples show how taxation thresholds affect a customer's bill. In the examples a tax rate of 5% and a threshold amount of \$21.30 is used.

Example 1 - Account With Taxes Under Threshold

This example shows the bill for an account where the accumulated tax amount is less than the threshold amount. Since the accumulated tax amount of \$10.50 is less than the threshold amount, taxes are not applicable and the account's bill should be adjusted to exclude the tax amount of \$10.50.

Bill Line	Amount
Standing Charge	10.00
2,000 units @ \$0.10	200.00
Tax @ 5% on \$210.00	10.50
Total	220.50
Adjusted Total	210.00

Example 2 - Account With Taxes Above Threshold

This example shows the bill for an account where the accumulated tax amount is greater than the threshold amount. Since the accumulated tax amount of \$27.92 is greater than the threshold amount, taxes are applicable and the account should be billed for the total amount.

Bill Line	Amount
Standing Charge	10.00
4,000 units @ \$0.10	400.00
Additional Charge	148.45
Tax @ 5% on \$558.45	27.92
Total	586.37

Example 3 - Account With Rounding Discrepancy

This example shows the resulting bill segment calc lines for an account with 3 service agreements. Here the accumulated tax amount at the bill level is \$21.29 with taxes calculated and rounded for each SA's bill segment. However, if the same taxes were calculated for each SA's bill segment and then accumulated and rounded at the bill level, the accumulated tax amount would be \$21.30 implying that taxes are applicable and the account should be billed for the tax amount of \$21.30 accounting for the discrepancy of \$0.01.

Bill Segment	Bill Line	Amount	Pre-rounding Amount
Bill segment for SA 1	964.70 units @ \$0.10	96.47	96.47000
	Tax @ 5% on \$96.70	4.82	4.82350
	Bill Segment Total	101.29	
Bill segment for SA 2	2222.90 units @ \$0.10	222.29	222.29000
	Tax @ 5% on \$222.29	11.11	11.11450
	Bill Segment Total	233.40	
Bill segment for SA 3	1072.40 units @ \$0.10	107.24	107.24000
	Tax @ 5% on \$107.24	5.36	5.36200
	Tax Discrepancy	0.01	
	Bill Segment Total	112.61	

NOTE:

Pre-rounding amount. Rate application captures two calculated amounts on the resulting bill segment calc lines. The first is the calculated amount rounded to two decimal places, and the second is a raw calculated amount with a five decimal precision. The base package algorithm that calculates taxation thresholds uses both amounts to account for any rounding discrepancy; however, only the raw calculated amount is used to compare against the taxation threshold.

Billing and Taxation Thresholds

The following sections describe how taxation threshold rules are implemented when a bill is produced.

Calculation of Taxation Thresholds

When taxation thresholds are applied at the account's bill level, it means that the system must calculate taxes for each of the account's service agreements, then sum these tax amounts and apply any applicable rounding rules. This accumulated tax amount is compared to a threshold amount and if the accumulated tax amount is less than the threshold amount, then taxes should not be applied to the customer's bill. To ensure the accumulated tax amount is accurate, all service agreements for an account whose tax amounts should be taken into consideration when comparing to the specified threshold should be billed together and the threshold comparison should take place after the bill segments have been generated. In addition, all bill segments whose tax amounts should be taken into consideration when comparing to the specified threshold must be for the same bill period.

NOTE:

Bill segments from different periods may not appear on the same bill. If a bill segment is canceled, all bill segments associated with that bill must be canceled and re-billed together. You cannot re-bill the bill segment on a bill for a different period.

An algorithm type is supplied with the base package that checks for taxation thresholds at bill completion time. The base bill completion algorithm type accumulates identified tax calc line amounts (accomplished using a bill segment calc line characteristic), and compares this to a specified threshold amount to determine if taxes apply for the account. If taxes do not apply, it sets a bill characteristic indicating this and regenerates the bill segment. Refer to the algorithm type [C1-CPBC-TAXT](#) for more information on how this type of algorithm operates.

NOTE:

Calculated adjustments are included in the evaluation against taxation thresholds. Adjustments that use a rate to calculate the adjustment amount may be included in the taxation threshold evaluation if their rate's components are set up to do so. The base algorithm [C1-CPBC-TAXT](#) looks at adjustments that are about to be swept onto the bill and, if applicable, includes them in the calculation.

The calculation rule(s) that calculate taxes make use of calculation rule eligibility criteria to ensure that taxes are only computed if the bill does not have the characteristic indicating that taxes are not applicable. Note that during the initial generation of each bill segment, the characteristic will not exist and taxes will be applied. This means that if you look at a bill before it is complete, taxes may not be accurately reflected.

Tax Amount Discrepancies

Since tax calculation rules are calculated and rounded at the bill segment level, it's possible that rounding discrepancies may occur if rounding of these tax amounts occurs at the bill level instead as illustrated in the example above. To account for this, the system uses both the two decimal precision and the five decimal precision calculated amounts that rate application captures on bill segment calc lines; however, only the raw calculated amount is used to compare against the taxation threshold. If there is a discrepancy in the tax amount (as shown in example 3 above), the system captures this amount as an entry in one of the bill segment's SQ collections prior to regenerating the bill segments. This is depicted in example 3 above where the tax discrepancy SQ resulted in an additional bill segment calc line on one of the bill segments for the rounding amount of 0.01. Note that a calculation rule is configured to bill for this discrepancy amount SQI.

NOTE:

Pre-rounding amount. Rate application captures two calculated amounts on the resulting bill segment calc lines. The first is the calculated amount rounded to two decimal places, and the second is a raw calculated amount with a five decimal precision. The base package algorithm that calculates taxation thresholds uses both amounts to account for any rounding discrepancy; however, only the raw calculated amount is used to compare against the taxation threshold.

Setting Up The System For Bill Taxation Thresholds

This section provides an overview of how to configure the system to calculate taxes at the account's bill level.

Installation Option

Select Freeze at Bill Completion as the bill segment freeze option. The system compares tax amounts calculated to a specified threshold amount and based on this determines if taxes should apply at bill completion time. If taxes should not be applied for the account, the system has to regenerate the bill segments as necessary.

Adjustment Types

Select Freeze at Bill Completion as the adjustment freeze option. The system compares tax amounts calculated to a specified threshold amount and based on this determines if taxes should apply at bill completion time. If taxes should not be applied for the account, the system has to regenerate the adjustments as necessary.

Bill Factors

Tax Threshold. Set up a separate bill factor for each distinct tax threshold amount.

Characteristic Type

Set up a characteristic type and value to identify the tax calculation rules and bill segment calculation lines that the system will use to compare to the specified threshold amount. Specify this characteristic type and value as parameters to the algorithms that apply the taxation threshold. This characteristic type and value must also be specified on each of your tax calculation rules that should be included in the threshold comparison.

Set up a characteristic type and value to use for the bill characteristic that indicates when bill level taxes apply. Specify this characteristic type and value as parameters to the algorithms that apply the taxation threshold.

Service Quantity Identifiers

Optional service quantity identifiers may be configured to capture the following:

- **Tax Amount.** The system compares the accumulated bill's tax amount to the specified threshold amount to determine if taxes are applicable. If taxes should not be applied for the account, the system regenerates the bill segments as necessary. If you wish to capture the tax amount computed for informational purposes, then a tax amount SQI should be set up and specified as a parameter on the algorithms that apply the taxation threshold.

- **Tax Discrepancy Amount.** Since tax calculation rules are calculated and rounded at the bill segment level, it's possible that rounding discrepancies may occur if rounding of these tax amounts should take place at the bill level. If you wish to account for these rounding discrepancies, then a tax discrepancy amount SQI should be set up and specified as a parameter on the algorithms that apply the taxation threshold. Your rates should also be configured to cater for this rounding discrepancy. The system adds an entry for the discrepancy amount to one of the bill segment's SQ collections prior to regenerating the bill segments.

Algorithms

Apply Taxation Threshold. Define an algorithm for each distinct tax threshold amount. Plug the algorithm(s) in on the appropriate customer class for the Pre Bill Completion system event. Refer to the [C1-CPBC-TAXT](#) algorithm type for more information about the base package algorithm.

Adjustment Generation - Apply Rate. Define an algorithm for each rate to be used by calculated adjustment types that are to be included in the tax threshold evaluation. Plug the algorithm(s) in on the appropriate adjustment type for the Generate Adjustment system event. Refer to the [ADJG-RT](#) algorithm type for more information about the base package algorithm.

Calculation Rules For Bill Taxation Thresholds

Your calculation rules that bill for taxes require the following:

- A characteristic that identifies them as calculation rules to include in taxation threshold comparisons
- Calculation rule eligibility criteria to ensure that taxes are not calculated if the taxation threshold algorithm dictates this

Only one eligibility group on the calculation rule is required. It would look as follows:

Group No.	Group Description	If Group is True	If Group is False
1	Tax applies if total accumulated tax amount at the account's bill level exceeds the threshold amount	Apply calculation rule	Skip calculation rule

The following criteria will be required for this group:

Seq	Field to Compare	Comparison Method	If True	If False	If Insufficient Data
10	Bill characteristic: Characteristic type = Tax Not Applicable indicator	= YES	Group is false	Check next condition	Group is true
20	Characteristic Collection: Characteristic type = Tax Not Applicable	= YES	Group is false	Group is true	Group is true

NOTE:

The second criterion included above is used to evaluate the applicability of the tax on calculated adjustments. Since these adjustments are not yet linked to the bill, the first criterion cannot be used to evaluate the applicability of the tax threshold. The pre-bill completion algorithm [C1-CPBC-TAXT](#) instead adds the characteristic to the characteristic

collection for use by the rate application when evaluating the adjustment's rate's eligibility rules. Refer to the [ADJG-RT](#) algorithm type for more information about the base package algorithm.

FASTPATH:

For more information, refer to [Designing Calculation Groups and Rules](#).

An additional calculation rule is required to bill for the tax rounding discrepancy as follows

- Calculation Rule type = Service Quantity
- UOM/TOU/SQI = the identifier of the service quantity containing the tax discrepancy amount
- Value Type = Unit Rate
- Value Source = Value
- Value = 1

Other Financial Transaction Topics

Various topics about financial transactions are discussed in this section.

The Source Of GL Accounts On Financial Transactions

The following table lists the major financial events, their standard accounting, and the source of distribution codes used to derive the GL accounts sent to your general ledger.

Financial event	GL Accounting	Source Of Distribution Code
Create a normal utility bill segment. Bill Segment FT Algorithm is Payoff Amt = Bill Amt / Current Amt = Amt Due	Debit: A/R	SA Type
	Credit: Revenue / Taxes Payable	Calculation Rule
Create a bill for company usage. Bill Segment FT Algorithm is Payoff Amt = 0 / Current Amt = 0	Debit: Company Usage Expense	SA Type
	Credit: Revenue / Taxes Payable	Calculation Rule
Create a bill for charity. Bill Segment FT Algorithm is Payoff Amt=0 / Current Amt = Bill Amt	N/A - charity bills have no effect in the GL	N/A
	N/A	N/A
Create a payment segment for a normal utility service agreement	Debit: Cash	Bank Account on the Tender Source of the Tender Control for the Payment Segment's Tender.
	Credit: A/R	SA Type
Create a payment segment for a charitable contribution service agreement	Debit: Cash	Bank Account on the Tender Source of the Tender Control for the Payment Segment's Tender.

	Credit: Charity Payable	SA Type
Create a payment segment for auto-pay at bill completion time	Debit: Cash	Bank Account on the Tender Source on the Auto-pay Route Type of the Auto-pay Source.
	Credit: A/R	SA Type
Canceling a payment	Debit: A/R	SA Type
	Credit: Cash	Bank Account specified by the user on the cancel tender page. Note that this defaults to the original tender's bank account.
Create an adjustment to levy a charge	Debit: A/R	SA Type
	Credit: Revenue	Adjustment Type

The bottom line is as follows:

- If a bill segment has a financial effect, the distribution code to debit comes from the distribution code on the SA Type, the distribution code to credit comes from the calculation rule(s) used to calculate the bill segment.
- Payment segments always have a financial effect; the distribution code to debit comes from the bank account on the tender source of the tender control of the tender, the distribution code to credit comes from the SA type.
- If an adjustment has a financial effect, the distribution code to debit and credit comes from the SA type and adjustment type. If the adjustment is positive (i.e., the customer owes your organization more money), the distribution code to debit comes from the SA type; the distribution code to credit comes from the adjustment type. Vice versa if the adjustment is negative.

Defining Customer Options

The definition of a customer is someone (or something) with financial obligations with your company. These obligations ensue because the customer has agreed to purchase goods or services at an agreed price.

You may be surprised to learn that there is no "customer" record in the system. Rather, the system subdivides customer information into the following records:

- **Person.** The person record holds demographic information about your customers and every other individual or business with which your company has contact. For example, in addition to customers, person records also exist for landlords, contractors, accountants at corporate customers, guarantors of customers, energy distributors, collection agencies, etc.
- **Account.** Accounts are the entities for which bills are produced and therefore you must create at least one account for every person who has financial obligations with your company. The account record contains information that controls when the bills are created and how the bills are formatted.
- **Service Agreement.** Think of a service agreement as a contract between your company and the customer. The service agreement contains the terms and conditions controlling how the bill details are created. Every account will have at least one service agreement (otherwise, nothing will appear on the account's bills).

Before you can define persons, accounts, and service agreements, you must set up the control tables defined in this section.

FASTPATH:

For more information about how persons, accounts and premises are used by your customer service reps, refer to [Understanding The "V"](#).

NOTE:

The tables in this section are only some of many tables that must be set up before you can bill your customers for the service(s) they consume. In this section, we limit the discussion to those tables that control basic demographic

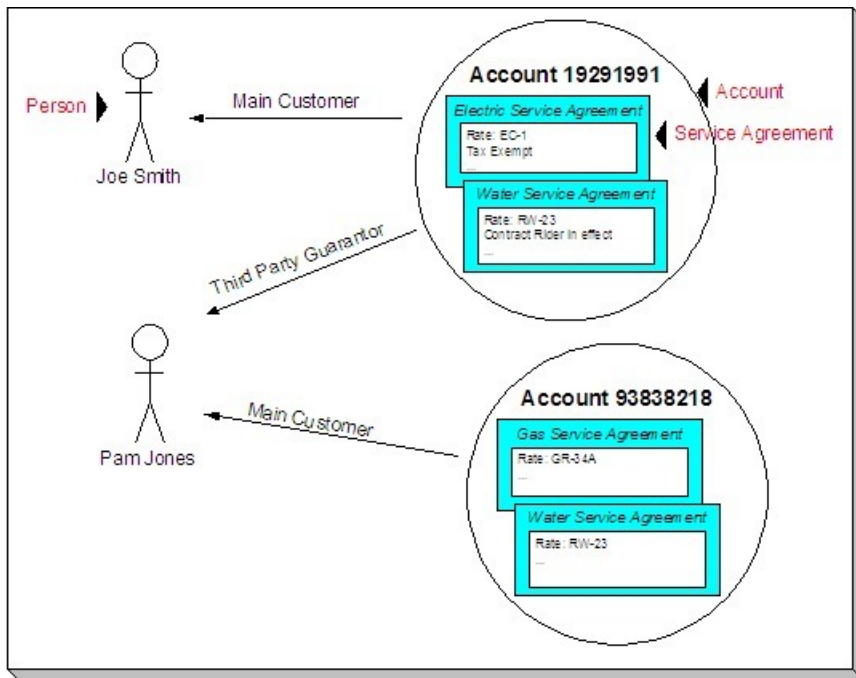
and financial information. In later sections, we describe the tables that control other billing-related functions like bill creation algorithms, meter reading and rates. It is only after all of these tables are set up that you will be able to generate bills and record payments.

Customer Overview

This section describes how the person, account, and service agreement records are used to record your customers' demographic and billing options.

A Simple Example Of Two Customers

The following picture illustrates two customers: Joe Smith and Pam Jones. Joe is the "main customer" on his account. Pam is the "main customer" on her account. Pam is also the "third party guarantor" on Joe's account.



Persons

Person records hold demographic information about the individuals and businesses with whom your organization communicates. Demographic information includes phone number(s), names and aliases, identification numbers, life support equipment needs, employment information, etc.

In the above example, 2 person records would be needed; one for Pam Jones and another for Joe Smith.

A new person is added when you first have contact with a person; the person does not have to be a customer before it is added. So, for example, if your company is starting a new marketing campaign, you can add information about potential customers the moment they are identified.

NOTE:

Businesses are persons too. In addition to humans, you use person records to maintain basic information about the businesses with which your organization has contact.

FASTPATH:

For a description of the control tables that must be set up before you can define a person, refer to [Setting Up Person Options](#).

Accounts

An account is analogous to an account at a bank:

- A person or business with no financial dealings with a bank will have no account (but the bank may choose to keep demographic information about the person as part of their marketing efforts). The exact analogy exists in this system.
- Individuals with financial dealings with a bank will have one or more accounts. The number of accounts is up to the customer. The exact analogy exists in this system.

A simple way to determine the number of accounts a customer will have is to ask "how many bills do they want each period?" because a customer receives one bill for each account. For example:

- A residential customer who also owns a small business may choose to receive two bills each month; one for the residence, the other for the business. This way, the charges for their business would be segregated from their personal charges. This customer would have two accounts.
- A conglomerate that owns several factories may want their transportation gas charges to appear on a single bill rather than have a separate bill for each factory. This customer would have a single account.

Account ID Is Non-Intelligent

The unique number of an account is referred to as the "account ID". You are probably very comfortable with this concept. You may, however, have difficulty dealing with the fact that the account id in this system has no intelligence built into it (e.g., many systems include the bill cycle and geographic location in the account id). In this system, the account ID is a random, system-assigned value.

Because the account ID contains no meaning, it can remain with a customer for life, regardless of where they live, when they are billed, the type of service they receive, etc. This is important because it means that all of the financial history linked to the account remains with the customer for life.

NOTE:

The non-intelligence of the account ID is also important from the perspective of the parallel processing that takes place when the system creates bills. Because the collection of accounts to be billed in any given bill cycle will be randomly distributed through the number spectrum, the system can distribute account number ranges to parallel threads and each thread will process roughly the same number of accounts.

Account / Person Cross-Reference

A person may be linked to zero or more accounts. A person won't be linked to an account when they have no financial relationship with your organization. A person will be linked to multiple accounts when they have financial relationships with more than one account.

An account must reference at least one person (i.e., the main customer), but may reference an unlimited number of individuals. Multiple persons are linked to an account when several parties have some type of financial relationship with the account (e.g., third party guarantors, account contact, bill copy recipients, etc.).

When Is An Account Created?

A person can exist without an account until such time as the person formally requests the commencement of service. The moment the customer requests service, an account must be created (and the person must be linked to the account).

When Is An Account Expired?

Accounts never expire. Once a customer has an account, the account remains in the system forever. Linked to the account are service agreements that define the price and conditions of a service supplied to the customer. When an account has active service agreements, the system produces bills for it. If the account doesn't have active service agreements, the system will not produce a bill for it. You can think of an account without active service agreements as being "dormant", waiting for the day when the customer again starts service. If the customer never restarts, the account (along with its financial history) remains dormant forever.

Service Agreements

A service agreement is a contract (either formal or implied) between your organization and a customer. Every service agreement contains the price and conditions of a service supplied to a customer.

A service agreement is linked to an account. There is no limit to the number of service agreements that may be linked to an account.

When Is A Service Agreement Created?

A service agreement is created when the customer requests service (not when service commences). Typically, service agreements are created in the pending state and field activities are generated to connect service. When the field activities are complete, the service agreement becomes active and the billing process starts generating bill segments for the service agreement.

FASTPATH:

For more information about starting service, refer to [The Big Picture Of Starting Service](#). For more information about bill segments, refer to [Bill Details](#).

Financial Transactions Are Linked To Service Agreements

FASTPATH:

For more information about how financial transactions are linked to service agreements, refer to [The Financial Big Picture](#).

When Is A Service Agreement Expired?

A service agreement is expired when the customer requests service be stopped. At that time, the service agreement is transitioned to the pending stop state and field activities are generated to stop service (these activities might involve simply reading the meter or they could involve disconnecting or removing the meter). When the field activities are complete, the system transitions the service agreement to the stopped state and the billing process generates a final bill for the service agreement. When the customer pays the final bill, the system transitions the service agreement to the closed state

FASTPATH:

For more information about stopping service, refer to [The Big Picture Of Stopping Service](#).

Setting Up Person Options

This section describes tables that must be set up before you can define persons.

Setting Up Identifier Types

When you set up a person, you may define the various types of identification associated with the person, e.g., their driver's license number, their tax identity, etc. Every piece of identification associated with a person has an identification type. These identifier type codes are defined using **Admin > Identifier Type**.

NOTE:

How are person identifiers used? The reason why identifiers are defined on a person is so that users you can look for a customer using one of their person identifiers (see [Control Central - Search Facilities](#) for more information). In addition, person identifiers help prevent duplicate persons from being added to the database. This is because the system warns a user before they add a new person when a person exists with the same identifier.

Person identifier types are optional. An [installation option](#) controls whether at least one identifier type is required on every person.

Description of Page

Enter an easily recognizable **ID Type** and **Description** for the Identifier Type.

If the identifier type has a format against which validation can be performed, use **Identifier Format** to define the algorithm. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that validates identifier types. Click [here](#) to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ID_TYPE](#).

Setting Up Person Relationship Types

It is possible to associate persons to other person. For example,

- You might want to define the subsidiaries of a parent corporation
- You might want to define spouses as separate persons and then link each person to another person

When you link a person to another person, you must define in what way the person is related to the other person by using a person relationship type code. These codes are defined using **Admin > Person Relationship Type**.

Description of Page

Enter the following for each relationship type:

- Enter an easily recognizable **Relationship Type** code.
- Use **Description (Person1=>Person2)** to describe how the first person is related to the second person.
- Use **Description (Person2=>Person1)** to describe how the second person is related to the first person.

NOTE:

Person1 versus Person 2. When you link persons together, you do it in respect of one of the persons (which we call Person 1). For example, if you want to link the subsidiaries to a parent company, you do this in respect of the parent company (i.e., you define the parent company's subsidiaries using the [Person - Persons](#) transaction).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PER_REL_TYPE](#).

Setting Up Contact Routings

Base values for Contact Routing are supplied. You can add your specific contact routings using the **Contact Routing** extendable lookup. You can add more contact routings, but the ones supplied with the system cannot be edited or deleted.

NOTE: Person contacts that reference the CSS routing type cannot be added by a user, and are not permitted to be edited or deleted. As well, records with this contact routing are only visible in the Person - Main page and are not displayed in the Account Portal or other places in the system.

Refer to [Defining Extendable Lookups](#) and for more information on lookups.

If your implementation is using notification preferences, you may need to set up delivery types before setting up contact routings. Refer to [Setting Up Delivery Types](#) for more information.

Setting Up Person Contact Status

Person contacts can have a status; for example, the status of an email contact could be pending, verified, or bounced. The product is delivered with statuses that are used in conjunction with opt-in functionality. When new person contacts are added, the initial status can be defaulted from the person contact type. See [The Person Contact Status Can Be Controlled By A Process](#), *System Person Contact Statuses* and *Opt-in Process* sections for more information.

Your implementation may choose to introduce other logic to update the status. See [The Person Contact Status Can Be Controlled By A Process](#) for more information. You can add your specific person contact statuses by updating the value for the CND_VERIF_STATUS_FLG lookup field.

When new person contacts are added, the initial status can be defaulted from the person contact type. Refer to the Person Contact zone's help text for more information.

Setting Up Person Contact Type Algorithms

Format Validation

Person contact information can be subject to format validation such as ensuring a phone number is entered in a particular format.

Click [here](#) to see the algorithm types available for this system event.

The base product provides the algorithms [C1-VALEMFMT](#) (Validate Email Format), [C1-VALPHFMT](#) (Validate Phone Format), and [C1-VALANYFMT](#)(Any Format Valid), which your implementation may use (and if so, no further configuration is needed). If your implementation's business rules require additional format validation algorithms, you may introduce your own.

NOTE: These algorithms confirm that a specific format exists and do not reformat or apply formatting to the corresponding field; for example, the *C1-VALEMFMT* (Validate Email Format) algorithm confirms that the email address contains required features (uch as an @ symbol and domain). It will not, however, insert these features.

If your implementation's business rules require additional format validation algorithms, you may introduce your own.

Setting Up Person Contact Types

Person contact types define the format for entering person contacts and information about the person contact.

To set up contact types, select **Admin menu > Person Contact Type > Add**.

The [Person Contact Type portal](#) contains the following zones:

- [Person Contact Type List](#)
- [Person Contact Type Zone](#)

Person Contact Type Portal

This section describes the zones associated to the Person Contact Type Portal:

- [Person Contact Type List](#)
- [Person Contact Type Zone](#)

Person Contact Type List

The Person Contact List zone lists all person contact types. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent person contact type.
- The standard actions of Edit, Delete and Duplicate are available for each person contact type.
- Click the Add link in the zone's title bar to add a new person contact type.

Person Contact Type Zone

The Person Contact Type zone contains display-only information about the Person Contact Type. This zone appears when a Person Contact Type has been broadcast form the Person Contact Type List zone or if this portal is opened via drill down from another page.

Please see the zone's help text for information about this zone's fields.

Where Used

Follow this link to open the data dictionary where you can view tables that reference [C1_PER_CONTDDET](#).

Choosing to Use Person Contact or Person Phone

Person contact is an updated and more configurable alternative to person phone and email. In this release, a feature is introduced that some system processes use to determine if person contact or person phone and email is being used. Person phone and email are referred to as legacy behavior. Legacy behavior is enabled when the Legacy Person Contact **Option Type** on the Customer Information Options [Feature Configuration](#) is set to Y. When set to N or not set legacy behavior is not enabled.

If your implementation uses functionality that has not been updated to use the feature, such as campaigns and order, you may wish to enable legacy behavior and continue to use person phone and email. It is important to note that Person Contacts are required for notifications. See [Maintaining Person Phone via Person Contact](#) for information about how person contacts can be created from person phone.

The following system processes have been updated to alter functionality based on whether legacy behavior is enabled or not.

System Process	Behavior When Feature is Not Set	Behavior When Feature is Set
Person	The Person Phone Grid and the Person Email field are not shown	The Person Phone Grid and the Person Email field are shown. Person Contact is always shown.
Person	The functionality to maintain person contacts from person phone is not triggered even when a person contact type is associated with a phone type.	The functionality to maintain person contacts from person phone is enabled.
Account/ Person	When an email or fax bill or quote route type is chosen, then information displayed is related to person contacts.	When an email or fax bill or quote route type is chosen, then information displayed is related to person phone or email.
Start/Stop	The Person Contact collection is shown and can be edited.	The Person Phone collection and Person Email are shown and can be edited.
Case	The Preferred Contact Methods associated with legacy person phone and email are not shown.	The Preferred Contact Methods associated with legacy person phone and email are available. Person Contact is always available.
Customer Contact	The Preferred Contact Methods associated with legacy person phone and email are not shown.	The Preferred Contact Methods associated with legacy person phone and email are available. Person Contact is always available.

Person Contact Status Can Be Controlled By A Process

Overview

Person contact types can be configured to allow a status. For some person contact types, your implementation may want to control the status through a process instead of allowing the status to be set manually. An example of this is using person contact status to capture opting in for receiving text messages, which is required in some jurisdictions. Opt-in is seeking permission from a customer to use a person contact for a purpose such as sending a customer text messages.

Opt-in works in conjunction with the functionality to enable opt-in for a delivery type. Each can function independently, but together they offer a complete solution. Since they can function apart from each other, each is described independently. See [Enabling Opt-in for a Delivery Type](#) for more detail on this logic.

To control a person contact status with a process, the person contact type must first be configured to allow status. The person contact type must then be set up in the opt-in section of the **Notification Preferences** [master configuration](#). By defining a Status Script, the person contact status can only be changed by the logic contained in or initiated from the script. The product is delivered with a script that creates an opt-in service task. This is designed specifically to support opting in to receiving notifications via text messages (SMS).

Opt-In Process

To use the delivered functionality, configure a new service task type for opt-in. This task type should reference the Notification Preference Opt-In Task Type Business Object (C1-NotifPrefOptInTaskType). The product is delivered with a service task business object. The service task is responsible for initiating an opt-in request, updating the person contact status, and logging the user response. An inbound web service receives responses and transitions the service task. These components are designed to integrate with Notification Center. When a customer is sent an opt-in request, they are asked to confirm. The content of the message explains the commands to the user. Examples are CONFIRM or STOP. Notification Center is configured to interpret the specific commands and map them to those expected by the inbound service, which transitions the service task appropriately.

Aside from the initial opt-in request, users can stop or unsubscribe from notifications at any time. The delivered solution supports two modes for this functionality: unsubscribing from all notifications for a particular person contact or unsubscribing from a specific notification type for a particular person contact. This is controlled through configuration, and relies on the external system such as Notification Center to determine the notification type.

The **Notification Preferences** [master configuration](#) help text contains detailed information about the configuration required to control person contact status and setting up opt-in processes.

It is important to note that while the delivered process to control person contact status was designed to work in conjunction with the opt-in process, your implementation can design a custom process that can create a service task or do something completely different. Also, it can be independent of opt-in and notifications. For example, the process can be used to verify that an email address is valid by sending the customer a verification request.

There is another concept, which is enabling opt-in for a delivery type. This is closely related to the functionality described here, but has different implications in the system. See [Enabling Opt-in for a Delivery Type](#) for more information.

There is a limitation worth noting. A person contact type may be set up to allow multiple delivery types, however the delivered solution only allows a single delivery type to be associated with the process used to automate person contact status. Without this limitation, different processes would be responsible for updating a single person contact status and those updates could be conflicting.

The System Person Contact Statuses

The opt-in solution delivered with the product utilizes the three delivered status values. They are: **Pending**, **Approved**, and **Rejected**. Your implementation can add your own person contact statuses, but they will not work with the solution delivered with the product. See [Setting Up Person Contact Status](#).

How the System Automates Person Contact Status

There are many ways that the process that controls person contact status can be invoked. For all of these, the person contact type must be set up in the **Notification Preferences** [master configuration](#) to have its status controlled by a process and, unless noted otherwise, the person contact is not already **Approved**.

- The primary way to initiate the process to control person contact status is when a person contact is used to add a contact preference and the contact preference's delivery type matches the delivery type associated with the person contact type as defined in the Notification Preferences master configuration. This method relies on enabling opt-in for a delivery type. This is how the two processes work with each other. See [Enabling Opt-in for a Delivery Type](#) for more information. An example of this scenario is that a person contact exists and is of a type that can receive text messages, such as "Mobile Phone". Until the person contact is used on a contact preference to receive a notification, there is no reason to initiate an opt-in process.
- This process can be automatically initiated by configuring a person contact to default to an initial status of **Pending**. When a person contact is added, the process is initiated. An example of this scenario is an email person contact initiating a process to verify the email address as soon as it is added to the system.
- When the value of an approved person contact is changed, the process is reinitiated. For example, even though the approval is attached to a specific person contact in the system, which has a unique ID, externally the approval was for

a particular phone number or email address. When the person contact value (i.e. the phone number or email address) is changed, the prior approval is no longer valid. The person contact status is changed to *Pending*.

A user can manually initiate the process. A button is provided on the Person Contact row on Person - Main. The button can be used to reinitiate the process when the customer does not have the request and needs it resent. It can also be used to manually initiate the process upfront - before the person contact is used on a contact preference.

Maintaining Person Phone via Person Contact

Person phone is referenced extensively throughout the system. At this time, some system processes have been updated to determine if phone data and email data is being maintained on person contacts or being maintained on person phone and email. See [Choosing to Use Person Contact or Person Phone](#) for more information.

To keep person phone records in sync with a person contacts, you should consider which phone types should be stored as contacts, configure the person contact type corresponding to each phone type, and update the phone type to reference the corresponding person contact type. Many phone types may be associated with a single person contact type, but a single phone type may only be associated with one person contact type.

The base product provides a batch process [C1-INPUS](#) (Create Person Contact from Person Phone/Email) that creates a person contact for all person phone records whose phone type is associated with a person contact type. The batch process can also create person contact records for the email address on the person. You would run this process to create person contact records if you already have phone or email data for persons, and you wanted to start leveraging person contact as the repository for phone and email information. See the batch process description for more information.

NOTE:

There is a primary switch on person contact records; one person contact for each unique contact routing must be set as the primary contact. When the batch process creates person contacts from person phone data, the first person contact added for each unique contact routing is set as primary. This can be controlled to some extent by your implementation by associating phone types with person contact types sequentially and running the batch process multiple times. For example, you may prefer that a home phone be the primary over a cell phone if both exist. To ensure this happens, associate the home phone with a person contact type and run the batch process. Then associate the cell phone with a person contact type and run the batch process again. This will force all home phone records to be processed first. It is recommended that regardless of how many times the batch processes run, they are run back to back. This should be thought of as a conversion of data and should be treated as such.

When a phone type is associated with a person contact type the following occurs:

- When a phone record is added, a corresponding person contact is added if one does not already exist.
- When a phone record is updated, corresponding updates are made to the person contact.
- When a phone record is deleted, the system tries to delete the corresponding person contact. All business rules for person contacts are enforced. If the person contact is associated with a contact preference, the person contact cannot be deleted, and in turn the phone record cannot be deleted.
- Person contacts associated with phone types cannot be deleted directly. The delete function is disabled.
- Contact information on person contacts associated with phone types cannot be updated directly. The contact information and extension are disabled.
- Person contacts whose person contact type is associated with a phone type cannot be added from the account contact information zone in the account portal. These need to be added through person phone.

There is no direct association between a person phone record and person contact (i.e. there are no foreign keys on either record pointing to the other). The system uses the association between phone type and person contact type and the specific phone number in order to find the associated person contact.

Your implementation may decide to use person contacts to store phone data and not associate them with phone types. Functionality related to the system use of person phone will not be available such as updating a phone number through an order.

If your implementation associates phone types with person contact types and wishes to remove this association, it is best to remove all phone types from all person contact types.

WARNING: Making changes to the association of phone type to person contact type can have serious implications for your data. Changing the person contact type associated to a phone type will likely result in difficulty maintaining both person phone and person contact data.

Setting Up Statement Construct Options

This section describes tables that must be set up before a statement construct can be set up for a person to begin receiving financial statements.

FASTPATH:

For more information, refer to [The Big Picture of Complex Statements](#).

Setting Up Statement Route Types

Statement route types define the method used to route statements to persons. To define a statement route type, open **Admin > Billing > Statement Route Type**.

Description of Page

Enter a unique **Statement Route Type**, **Description** and **Statement Routing Method** for every statement route type.

NOTE:

The values for Statement Routing Method are customizable using the Lookup table. This field name is STM_RTG_METH_FLG.

The next two fields control how statements that are routed using this route type are printed (both in batch and online). Refer to [Technical Implementation Of Batch Statement Production](#) for more information about producing statements in batch. Refer to [Technical Implementation Of Online Statement Production](#) for more information about online statement production.

- Use **Batch Control** to define the process that creates the flat file that is passed to your statement printing software. If you use an **Extract Algorithm** to construct the downloaded information, you can use the STMDWLD process.
- Use **Extract Algorithm** to define the plug-in component that constructs the "flat file records" that contain the information to be merged onto statements routed using this route type. This algorithm is called when a user requests an online image of a statement on [Statement - Main](#) and it may also be called by the batch statement extraction process defined above. Click [here](#) to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_STM_RTE_TY](#).

Setting Up Account Options

This section describes tables that must be set up before an account can receive a bill.

Setting Up Account Management Groups

Users are informed that something requires their attention by entries that appear in To Do lists. For example, consider what happens when billing can't find a reading (and it's not allowed to estimate):

- The billing process creates a bill segment that is in error - meter read cannot be found.
- This error bill segment, in turn, triggers the creation of a To Do entry.
- The To Do entry is addressed to a role. A role is one or more users who can "action" the To Do entry.
- When a user views their To Do entries, they see all entries addressed to all roles of which they are part.

You can optionally use account management groups (AMG) to define the respective role to be assigned to To Do entries that are associated with an account and a given To Do type. For example, you can create an AMG called Credit Risks and assign this to accounts with suspect credit. Then, whenever an account-oriented To Do entry is created for such an account, it will be assigned a role based on the Credit Risks AMG. Refer to [Assigning A To Do Role](#) for more information..

NOTE:

Account management groups are optional. You need only set up account management groups (and link them to accounts) if you wish to address specific To Do entries associated with specific accounts to specific roles.

Account management groups are defined using **Admin > Customer > Account Management Group > Add.**

Description of Page

Enter an easily recognizable **Account Management Group** code and **Description** for each account management group. Use the grid to define the **To Do Role** to be assigned to entries of a given **To Do Type** that are associated with accounts that reference the **Account Management Group**.

NOTE:

Only To Do entries that are account-oriented take advantage of the roles defined for an account management group (because only accounts reference an account management group).

Where Used

Follow this link to view the tables that reference [CI_ACCT_MGMT_GR](#) in the data dictionary schema viewer.

Setting Up Account Relationship Codes

When you link a person to an account, you must define in what way the person is related to the account by using an account relationship code. These codes are defined using **Admin > General > Account Relationship Type.**

Description of Page

Enter an easily recognizable **Relationship Type** and **Description** for each relationship type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ACCT_REL_TYP](#).

Setting Up Alert Types

Account based alerts that appear in control central have an **AlertType**. To define valid alert types, navigate to **Admin > Customer > Alert Type**.

Description of Page

Enter an easily recognizable **Alert Type Code** and **Description** for each alert type. Specify the **Alert Days** to indicate the amount of time that alerts of this type will be effective by default. Specify a value of zero to indicate that alerts of this type will be effective indefinitely by default.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ALERT_TYPE](#).

Setting Up Bill Messages

There are various informational and warning messages that may appear on an account's bills. Each message is identified with a bill message code. To define a bill message code, open **Admin > Billing > Bill Message > Add**.

Description of Page

Enter a unique **Message Code** and **Description** for every bill message.

The following attributes control how and where the bill message appears on the customer's bill:

Priority controls the order in which the message appears when multiple messages appear on a bill.

NOTE:

The values for this field are customizable using the Lookup table. This field name is MSG_PRIORITY_FLG.

Insert Code controls whether a document should be inserted into the bill envelope when the bill message appears on a bill.

Message on Bill is the actual verbiage that appears on the customer's bill. If the message text is not static (e.g., field values need to be substituted into the body of the message), you can use the % *n* notation within the **Message on Bill** to cause field values to be substituted into a message. Refer to [Substituting Field Values Into A Bill Message](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BILL_MSG](#).

Setting Up Bill Route Types

Bill route types define the method used to route bills to accounts. To define a bill route type, open **Admin > Billing > Bill Route Type**.

Description of Page

Enter a unique **Bill Route Type** and **Description** for every bill route type.

Bill Routing Method controls the type of information that may be defined when the respective **Bill Route Type** is selected on [Account - Person Information](#). The following options are available:

- Postal: Use this method if the routing is via the postal service.
- Fax: Use this method if the routing is via fax.
- Email: Use this method if the routing is via email.

NOTE:

The values for **Bill Routing Method** are customizable using the [Lookup](#) table. This field name is BILL_RTG_METH_FLG.

The next two fields control how bills that are routed using this method are [printed](#) (both in batch and online).

- Use **Batch Control** to define the background process that performs the actual download of the billing information. Refer to [Technical Implementation of Printing Bills In Batch](#) for more information about these processes.
- Use **Extract Algorithm** to define the algorithm that constructs the records that contain the information that appears on a printed bill. Refer to [Printing Bills](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BILL_RT_TYPE](#).

Setting Up Bill Cycles

FASTPATH:

Refer to [Defining Bill & Service Cycles](#) for a description of how to set up bill cycles.

Setting Up Customer Classes

When you set up an account, you must assign it a customer class. The topics in this section describe the customer class control table.

Customer Class - Main

To set up customer classes, navigate to **Admin > Customer > Customer Class > Add** and use the **Main page** to define your Customer Class.

Description of Page

Enter a unique **Customer Class** code and **Description** for every customer class.

Use **Collection Class** to define the collection class that defaults onto new accounts that belong to this customer class. An account's collection class may be subsequently modified if the account has special collection problems or needs.

FASTPATH:

For more information about the significance of collection class, refer to [Designing Your Collection Classes](#).

Turn on **Business Activity Required** if service agreements linked to accounts with this customer class require a Business Activity description to be entered.

Turn on **Open Item Accounting** if accounts belonging to this customer class are subject to open-item account. Refer to [Open Item Accounting](#) for a complete explanation of the significance of this switch.

Turn on **Non CIS Payment** if accounts belonging to this customer class are used for payments made to reduce non-CIS debt. For example, assume your company accepts payments for a county assessor and you don't want to set up a separate account for each person who pays their assessment bill. You should set up the following information to accept such payments:

- Create a new customer class called "Non CIS Customer".

- Create a SA type for each type of non-CIS payment that customers can make. Make sure to enter a distribution code on each SA type that references the appropriate revenue (or payable) account. Don't forget to indicate that each SA type is not billed.

NOTE:

. Payment Templates can be used for common types of non-CIS payment allocations. These templates are used to default the payment distribution and allow non-CIS payments to be directly allocated to specific distribution codes.

FASTPATH:

For more information about using Payment Templates to process non-CIS payments, refer to [Non-CIS Payments](#).

- Create an account to which you'll book such payments. Have this account reference the new customer class. We recommend creating a separate account for each SA type that you created in the previous step.
- Create and activate a service agreement for the new account(s).

When someone pays for non-CIS debt, the operator will add a payment for the above account. On the payment, the operator should record reference information in order to know exactly why the payment was made. Refer to [Payment Event - Main](#) for more information.

You must define a variety of business rules for every division in which a customer class has customers. For example, if you operate in both California and Nevada AND you have CIS divisions for each state AND you have residential customers in each state, you must define **Customer Class Controls** for each CIS division. You do this on the [Customer Class - Controls](#) page. The grid that follows simply shows the CIS divisions for which business rules have been set up.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CUST_CL](#).

Customer Class - Bill Messages

When a customer class has bill messages, the system will sweep these messages onto bills created for accounts belonging to the customer class. Use this page to define a customer class's bill messages. Navigate to **Admin > Customer Class > Search** and then navigate to the **Bill Messages** page to maintain this information.

Description of Page

Use the bill messages collection to define **Bill Message** codes that should appear on bills that created for accounts that belong to a given customer class. For each message, also specify the **Start Date** and **End Date** when such a message should appear on the bill (leave **End Date** blank if the message should appear indefinitely).

Where Used

The system snaps customer class bill messages on a bill during bill completion. For more information about bill messages, refer to [The Source Of Bill Messages](#).

Customer Class - Controls

You must define a variety of business rules for every division in which a customer class has customers. For example, if you operate in both California and Nevada AND you have CIS divisions for each state AND you have residential customers in each state, you must define **Customer Class Controls** for each CIS division in respect of the residential customer class. Open **Admin > Customer > Customer Class > Search** and then navigate to the **Controls** page to maintain this information.

Description of Page

The **Customer Class Controls** scroll contains business rules governing accounts that belong to a **CIS Division** and **Customer Class**. The following fields should be defined for each **CIS Division**:

- Use **Days Till Bill Due** to define the number of days after the bill date that the customer's bill is due. If the due date is a weekend or company holiday, the system will move the due date forward to the next workday (using the workday calendar defined on the account's CIS division).
- Specify the **Budget Plan** that defaults onto new accounts belonging to this customer class. Please note that an account's budget plan may be subsequently modified if the account has special budget processing needs. Refer to [Setting Up Budget Plans](#) for more information.
- Use **Min Credit Review Freq (Days)** to define the maximum number of days that can elapse between the reviews of an account's debt by the [account debt monitor](#). Note, a value of zero (0) means that accounts in this customer class will be reviewed every day.
- Use **Credit Review Grace Days** to define the number of days after the bill due date that an account should be reviewed by the [account debt monitor](#).
- Turn on the **Late Payment Charge** if customers in the class / division combination are eligible for late payment charges.
- Use **LPC Grace Days** to define the number of days after a bill's due date that a late payment charge will be generated (if the various LPC algorithms allow such - refer to [How Late Payment Charges Get Calculated](#) for the details). If the grace date falls on a weekend or holiday, the system moves the grace date to the next available workday (using the workday calendar defined on the account's CIS division).

The grid that follows contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

WARNING:

These algorithms are typically significant system processes. The absence of an algorithm may prevent the system from operating correctly.

You can define algorithms for the following **System Events**:

System Event	Optional / Required	Description
Autopay Amount Over Limit	Optional	<p>This algorithm is called to handle the situation when a system-initiated automatic payment is created that exceeds the customer's maximum withdrawal limit. Specifically, this algorithm is called when:</p> <ul style="list-style-type: none"> - The account has a maximum withdrawal limit on their automatic payment options - The system attempts to create an automatic payment that exceeds this amount - The automatic payment algorithm that's plugged into the installation record has logic that invokes this algorithm when the above conditions are true <p>If you do not plug-in this type of algorithm and the above situation is detected, the automatic</p>

payment will be created and no error will be issued.

Refer to [How To Implement Maximum Withdrawal Limits](#) for more information.

Click [here](#) to see the algorithm types available for this system event.

Bill Cancel	Optional	<p>This algorithm provides the ability to include additional cancel logic when canceling online.</p> <p>Algorithms of this type can be called in two modes: D (Determine Bill Page Buttons) and X (Cancel Bill). Mode 'D' governs whether an action button to cancel the bill will appear on the Bill page and mode 'X' performs the actual cancellation logic.</p> <p>Click here to see the algorithm types available for this system event.</p>
Bill Completion	Optional	<p>When a bill for an account is completed, bill completion algorithms are called to do additional work.</p> <p>Refer to the description of the Complete button under Bill Lifecycle for a description of when this algorithm is called during the completion process.</p> <p>Click here to see the algorithm types available for this system event.</p>
Bill Eligibility	Optional	<p>Algorithms for this plug-in spot are called when generating a bill in batch billing. It provides the ability to determine if an account is ineligible for billing and should therefore be skipped from further processing.</p> <p>If an eligibility algorithm is not used, a bill is created for any account in the open bill cycle and is later deleted by the billing process if it detects that there is no information linked to the bill.</p> <p>Click here to see the algorithm types available for this system event.</p>
Bill Segment Freeze / Cancel	Optional	<p>When a bill segment for an account in this customer class / division is frozen or canceled, an algorithm of this type may be called to do additional work.</p> <p>Refer to Bill Segment Lifecycle for more information about freezing and canceling bill segments.</p> <p>Click here to see the algorithm types available for this system event.</p>
FT Freeze	Optional	<p>When an FT is frozen, this algorithm is called to do additional work.</p>

For example, if you practice [Open Item Accounting](#), you will need such an algorithm to handle the cancellation of match events when a financial transaction is canceled that appears on a match event. Refer to [How Are Match Events Cancelled?](#) for more information about cancellation.

Click [here](#) to see the algorithm types available for this system event.

Late Payment Charge Eligibility

Required if the customer class / division is eligible for late payment charges

This algorithm is called by the late payment process to determine eligibility for late payments.

Just because an account's customer class allows late payment charges to be calculated doesn't mean the account's delinquent service agreements will be levied late payment charges. In addition, a delinquent service agreement's SA type must reference a late payment charge algorithm. Refer to [SA Type - Main](#) for more information about SA type late payment charge issues. Refer to [How Late Payment Charges Get Calculated](#) for more information about late payment charges in general.

NOTE:

Only One Algorithm. Only one late payment charge eligibility algorithm may be defined for a customer class / CIS division combination.

Click [here](#) to see the algorithm types available for this system event.

Levy an NSF Charge

Optional

This algorithm is called when a payment is canceled with a cancellation reason that indicates an NSF.

Refer to [NSF Cancellations](#) for more information about what happens when a payment is canceled due to non-sufficient funds.

NOTE:

Only One Algorithm. Only one algorithm to levy an NSF charge may be defined for a customer class / CIS division combination.

Click [here](#) to see the algorithm types available for this system event.

Order Completion

Optional

When an [order](#) is completed for a customer linked to this customer class, this algorithm is called to do additional work (e.g., create a customer contact). You need only specify this type of algorithm if you require additional work

to be performed when an order is completed for customers who belong to this customer class.

Click [here](#) to see the algorithm types available for this system event.

Overpayment Distribution	Required	<p>When a customer pays more than they owe, this algorithm is called to determine what to do with the excess funds. Refer to Overpayment Segmentation for a description on how to configure the system to handle your overpayment requirements.</p> <p>NOTE: Only One Algorithm. Only one overpayment distribution algorithm may be defined for a customer class / CIS division combination.</p> <p>Click here to see the algorithm types available for this system event.</p>
Override Due Date	Optional	<p>An account's bill due date will be equal to the bill date plus its customer class' Days Till Due. If you need to override this method for accounts in a specific customer class, specify the appropriate algorithm here.</p> <p>NOTE: Only One Algorithm. Only one due date override algorithm may be defined for a customer class / CIS division combination.</p> <p>Click here to see the algorithm types available for this system event.</p>
Payment Cancellation	Optional	<p>Algorithms of this type are called when a payment is canceled.</p> <p>Click here to see the algorithm types available for this system event.</p>
Payment Distribution	Required	<p>This algorithm is called to distribute a payment amongst an account's service agreements. Refer to Payment Distribution for more information about how payment distribution works.</p> <p>NOTE: Only One Algorithm. Only one payment distribution algorithm may be defined for a customer class / CIS division combination.</p> <p>Click here to see the algorithm types available for this system event.</p>
Payment Freeze	Optional	<p>When a payment is frozen, this algorithm is called to do additional work. If you practice</p>

[Open Item Accounting](#), you will need such an algorithm to link the payment's financial transactions to the match event that was originally created when the payment was distributed. Refer to [Payments and Match Events](#) for more information.

Click [here](#) to see the algorithm types available for this system event.

Post Bill Completion

Optional

When a customer class has algorithms of this type, they are called after the completion of a bill for an account linked to this customer class.

Refer to the description of the Complete button under [Bill Lifecycle](#) for a description of when this algorithm is called during the completion process.

Click [here](#) to see the algorithm types available for this system event.

Pre Bill Completion

Optional

When a customer class has algorithms of this type, they are called immediately before completion starts for an account linked to this customer class. These algorithms have the potential of:

- Deleting a bill. You might want a pre completion algorithm to delete a bill if a condition is detected that should inhibit the sending of a bill to a customer (e.g., the bill just contains information about recent payments).
- Aborting the completion process and creating a bill exception. If the algorithm indicates this should be done, the bill is left in the pending state and a bill exception is created describing why completion was aborted. You might want a pre completion algorithm to do this if, for example, integrity checks detect there is something wrong with the account or its service agreements. If the integrity check fails, the bill can be left in the pending state and a bill exception created describing why.

Refer to the description of the Complete button under [Bill Lifecycle](#) for a description of when this algorithm is called during the completion process.

Click [here](#) to see the algorithm types available for this system event.

Quote Completion

Optional

When a [quote](#) is completed for a customer linked to this customer class, this algorithm is called to do additional work (e.g., create a

		customer contact). You need only specify this type of algorithm if you require additional work to be performed when a quote is completed for customers who belong to this customer class. Click here to see the algorithm types available for this system event.
Write Off Method	Required if you allow users to write-off debt real time using the write off transaction	When a user presses the create button on the write off transaction , this algorithm is executed to write-off the selected debt. Refer to The Ramifications of Write Offs in the General Ledger for more information. Click here to see the algorithm types available for this system event.

Setting Up Collection Classes

FASTPATH:

Refer to [Setting Up Collection Classes](#) for a description of how to set up collection classes.

Setting Up Customer Information Options

When you add a new person, the system is set up by default to add an account for the person and go to the Start Service page when you save the new person information. You can change this functionality by configuring the following option types on the Customer Information Options feature configuration:

- **Add Account and Start Service Default:** Indicates whether the Add Account and Start Service option is selected on the person page, by default. If this option is not configured, the Add Account and Start Service option is selected by default.
- **Post Add Person BPA Script:** Indicates a BPA script to invoke when a user successfully adds or changes a person on the person page.

When you use control central to search for accounts, the system limits your search results based on your access rights. You can change this functionality by configuring the following option type.

- **Search All Accounts:** Indicates whether a Control Central search should allow a user to search all accounts without validating the user's access rights. If the user tries to select an account without having the required access, they will not be able to navigate to the Account Information tab on Control Central for the selected account.

For more information about Feature Configurations, see [Defining Feature Configurations](#).

Setting Up Customer Contact Options

This section describes tables that must be set up before you can define customer contacts.

FASTPATH:

Refer to [The Big Picture Of Customer Contacts](#) for more information about customer contacts.

Setting Up Customer Contact Classes

Every customer contact record has a contact type that classifies the record for reporting purposes. And every contact type, in turn, references a customer contact "class". The class categorizes customer contacts into larger groupings for reporting purposes.

Open **Admin > Customer > Financial > Customer Contact Class** to define your customer contact classes.

Description of Page

Enter a unique **Contact Class** and **Description** for each customer contact class.

After you have created your customer contact classes, you'll be ready to setup your [customer contact types](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CC_CL](#).

Setting Up Customer Contact Types

Every customer contact record has a contact type that controls the behavior of the customer contact.

FASTPATH:

Refer to [The Big Picture Of Customer Contacts](#) for more information about customer contacts.

Open **Admin > Customer Contact Type > Add** to define your customer contact types.

Description of Page

Every customer contact type is identified by a unique combination of **Contact Class** and **Contact Type**.

Enter a brief **Description** of the customer contact type.

Only specify a **Contact Shorthand** if customer contacts of this type can be added in the [Customer Contact Zone](#). The value you specify in this field is what the user selects to add a customer contact in this zone.

Use **Contact Action** if something should be triggered when customer contacts of this type are added. The only valid value in this release is Send Letter. If you select this option, you must also specify a **Letter Template**. Refer to [Printing Letters](#) for more information about how letters are produced.

Use the **Customer Contact Type Characteristics** collection to define characteristics that can be defined for contacts of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on customer contacts of a given type. Turn on the **Default** switch to default the **Characteristic Type** when customer contacts of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CC_TYPE](#).

Setting Up Notification Preference Options

The topics in this section provide background information about notification preferences.

Notification Preferences Overview

The Notification Preferences functionality is comprised of two main components: sending messages out of the system and maintaining customers' communication preferences for receiving these messages.

Notifications are created by an algorithm plugged into the Create Notification plug-in spot on the Notification Type (BO). Click [here](#) to see the algorithm types available for this system event. The base product provides the [C1-CRE-NTF](#) algorithm, which creates the notification via an outbound message.

The following sections describe specific notification preference concepts and how to configure the system to use notification preferences.

Push vs. Subscription Notification Types

Notification types are defined as either push – which means that a customer need not sign up to receive notifications of this type, or subscription – which means that a customer must sign up to receive this type of notification. As you can tell, they have different uses and different behaviors.

Push notification types allow users to specify communication preferences for a general type of notification. These preferences include contact preferences and opt-out preferences, which are applied to individual notifications belonging to the general type. For example, “marketing” may be the general type, and the individual notification types may include several different types of marketing notifications that contain different messages and are targeted at different channels. Refer to [Parent vs. Individual Push Notification Types](#) for more information.

In the absence of any contact preference or opt-out preference, push-based notifications can determine a default contact and send the notification using it. The logic to determine a default contact is defined via an algorithm plugged into the Determine Default Contact plug-in spot on the Notification Type (BO). Click [here](#) to see the algorithm types available for this system event. The base product provides the [C1-NTF-DFPRF](#) (Determine Default Contact for Notification Type) algorithm, which looks to retrieve a default contact from the account's main person.

Subscription notification types define a particular notification and its message. Customers must have an active contact preference to receive these notifications. Subscription-based notification types can use an underlying service task. Refer to [Some Notification Types Use Service Task](#) for more information. A subscription-based notification type can also be set up to allow routing of bills or quotes to multiple destinations.

Parent vs. Individual Push Notification Types

Push notification types are configured as either a parent or individual notification type. An individual notification type references a parent notification type; a parent cannot reference another parent. Customers define communication preferences for parent push notification types. Notifications are sent out for individual push notification types. The preferences defined for parent push notification types are applied to the associated individual push notification types.

This means that customers do not have to supply their preferences for every individual notification that your implementation may send. Storing customer preferences for a parent notification type permits those preferences to be inherited for individual notification types added later in time under the same parent. The parent push notification types can be thought of as categories of notifications for which your customers define contact preferences and opt-out preferences.

When system processes or your custom processes attempt to send out a push notification, it should be for an individual notification type. The system will retrieve the parent notification type from the individual notification type when checking for communication preferences.

One function available to push-based notifications as described in [Push vs. Subscription Notification Types](#) is the ability to determine a default contact in the absence of a contact preference or opt-out preference. If your implementation wants to take advantage of this function, but would like the customer to supply preferences for receiving an individual notification, a parent push notification type can be set up with only one individual push notification type.

Delivery Type

Delivery Type is the channel used to send a notification. Examples are email, text message (SMS), fax, and outbound IVR. Delivery type is used throughout notification preferences. If your implementation is not using notification preferences, you would not utilize delivery type.

- Notification types list the supported delivery types. For example, a bill ready notification type may support email and SMS while a notification type to encourage customers to enroll in a budget plan may only support email. The message subject and body are configured for each notification type's supported delivery type.
- Person contact types list the allowable delivery types. For example, a cell phone person contact type may allow SMS while a home phone person contact type may not allow any delivery types. An email person contact type would allow an email delivery type unless the person contact type exists to capture a customer's email address and not to be used in conjunction with notification preferences. Note: the delivery types that can be specified on a person contact type are restricted to those defined on the person contact types contact routing. Refer to [Setting Up Contact Routings](#) for more information.
- A contact preference for a customer consists of the combination of notification type, delivery type, and person contact.

Enabling Opt-In for a Delivery Type

Overview

Opt-in can be enabled for a delivery type and is configured using the [Notification Preferencesmaster configuration](#); for example, opt-in can be enabled for SMS, but not email.

To help clarify some terminology, as explained in [Person Contact Status Can Be Controlled by a Process](#), the product is delivered with a process that initiates a service task to request opt-in or permission to use a person contact. This solution requires that opt-in is enabled for the delivery type associated with the person contact type. In this regard, the two concepts are intertwined, but enabling opt-in for a delivery type can function independently.

Enabling opt-in is centered on the person contact status, while the process that controls the status can vary; therefore, it is possible to use enable opt-in for a delivery type while manually controlling the person contact status. Enabling opt-in for a delivery type has two key impacts system behavior.

- When a contact preference is added and the delivery type is enabled for opt-in, there is logic that checks if the person contact associated with the contact preference is in the **Approved** status. If it is not, and the person contact type is associated with a process to control the status, that process is initiated. If the person contact type is not associated with a process to control its status, a To Do Entry is created. The To Do Type is configured on the [Notification Preferences](#) master configuration. The system is delivered with To Do Type Opt-In Errors (**C1-OPTIN**) for this purpose.
- Notifications will not be delivered to person contacts that are not **Approved**. When a system process initiates an outbound notification based on an active preference, if the person contact associated with the contact preference is not **Approved**, a flag is set in the notification indicating that it should not be delivered. It is up to the receiving system, such as Notification Center, not to deliver the notification based on this flag. This is designed so that a record of the notification being generated is still captured. Also the [C1-NTF-DFPRF](#) (Determine Default Contact for Notification Type) will not choose a person contact that is not approved.

The [Notification Preferencesmaster configuration](#) help text contains detailed information about the configuration required to enable opt-in for a delivery type.

Some Notification Types Use Service Task

Prior to the introduction of notification preferences functionality, the system supported notifications set up through customer self-service. These notifications used service tasks to store customer preferences and in some cases trigger

notifications. The destination information for the notification existed outside of the system. Notification preferences, and more particularly communication preferences, define the user's preferences on how to be notified for each notification type and where the notification is to be sent. This reduces, but does not eliminate, the need to use a service task to store a customer's preference.

There are four existing notification types that rely on service tasks to trigger a notification. Additionally, one of the prepaid notifications uses a service task type business object with specific fields for that notification type. Refer to [Designing Notification Types](#) for information regarding the configuration requirements for certain notification types.

Notification Preferences supports configuring notification types with or without a service task and only subscription-based notification types can be configured with an underlying service task.

Designing Notification Types

Some notification types require a specific configuration in order for the system to create notifications for that type.

Service Task Based Notification Types

There are system processes that rely on a hard-coded notification type lookup, referred to as notification type (legacy) on the notification type configuration. The system identifies the user-defined notification type from the look of value.

The following identifies the configuration requirements for the notification types that are supported by existing processes in the system that rely on service tasks.

Notification Type (User Defined)	Push/Subscription	Controlled By	Notification Type (Legacy)	Service Task Type
Prepaid billing new charge	Subscription	Service Task	C1PC	Self-Service Task with BO C1-NotifyTaskType
Prepaid billing payment request	Subscription	Service Task	C1PP	Self-Service Task with BO C1-PPBPaymentNotifyTaskType
Payment Received	Subscription	Service Task	WSPR	Self-Service Task with BO C1-NotifyTaskType
Bill Ready	Subscription	Service Task	WXBR	Self-Service Task with BO C1-NotifyTaskType
Bill Due	Subscription	Service Task	WXBD	Self-Service Task with BO C1-NotifyTaskType
Late Payment	Subscription	Service Task	WSLP	Self-Service Task with BO C1-NotifyTaskType

Non-Service Task Based Notification Types

The following identifies the configuration requirements for the notification types that are supported by existing processes in the system that rely on the user-defined notification type to be configured.

Notification Type (User Defined)	Push/Subscription	Controlled By	Notes
Marketing Preference (Lead Event)	Sub of Ind. Push	N/A	Notification type configured on lead event type of BO C1-LeadEvtTypeNotificationPref
Forms Update	Sub of Ind. Push	N/A	Notification type configured on form task type of C1-FormTaskTypeForNotification

Bill and Quote Routing

Notification types can also be used to define contact preferences for bill and quote routing for email and fax. This can enable a bill or quote to create a routing for each active contact preference. For example for a single person on an account that receives a copy of the bill, the bill can be sent to more than one email address.

Some specific configuration is needed to support this.

The notification types are to be set up as follows:

Notification Type (User Defined)	Push/Subscription	Controlled By	Bill/Quote Route Type
Bill Route	Subscription	Bill Route	Specify the Bill Route Type
Quote Route	Subscription	Quote Route	Specify the Quote Route Type

Configure only the delivery type that corresponds with the bill or quote route's routing method. For example, if the bill route type's routing method is email, configure the email delivery type on the notification type.

Configure only person contact types whose contact routing corresponds with the bill or quote route's routing method. For example, if the bill route type's routing method is email, configure only person contact types with a contact routing of email. Ensure that the person contact type allows only the delivery type specified on the notification type.

This very specific configuration is needed because the system will override the existing logic that creates bill and quote routings when a notification type exists as described above and there are active contact preferences for that notification type with the specified delivery type and the person contact's person contact type.

Setting Up Notification Type Algorithms

Suppression Criteria

Notification types can be suppressed from appearing in a list for a particular customer. This is done by plugging in a suppression criteria algorithm on the notification type.

Click [here](#) to see the algorithm types available for this system event.

The base product provides the **C1-ALWY-SUP** (Always Suppressed) algorithm, which suppresses the notification type for all customers. This does not impact active contact preferences for the notification type, nor does it prevent processes from creating notifications of this type. This should be used with business process changes to stop sending notifications of this type. This can also be used temporarily after a notification type has been configured, but before it should be used.

The base product provides the **C1-CKSP-PPB** (Check Prepaid Biller Suppression) algorithm, which suppresses the notification type if the customer does not have an active prepaid biller task. This should be used on prepaid related notification types so that only customers on prepaid billing receive these notifications.

If your implementation's business rules require additional notification type suppression rules, you may introduce your own algorithm.

Override Do Not Disturb

Do not disturb information can be captured on person contacts. Person contacts are associated with a notification type to define a contact preference. When notifications are sent out, do not disturb information is included in the message to the receiving system. Some notification types might be important enough to not honor the customer's do not disturb information. For example, a notification might be sent to a subset of customers to warn of a gas leak and evacuation and needs to be sent regardless of the customers do not disturb wishes. This is done by plugging in an override do not disturb algorithm on the notification type. Note: this does not actually override the customer's information, but sends an indicator in the message to the receiving system that the do not disturb information should be overridden or ignored.

Click [here](#) to see the algorithm types available for this system event.

The base product provides the **C1-OVR-DND** (Override Do Not Disturb) algorithm, which returns a value of true that is passed to the receiving system in the notification.

If your implementation has more specific business rules related to overriding a customer's do not disturb preference, you may introduce your own algorithm.

Setting Up Delivery Types

Base values for Delivery Type are supplied. You can add your specific delivery types by updating the value for the C1_SS_DELIVERY_TYPE lookup field.

Setting Up Notification Types

Each notification type defines the controls and behavior for defining communication preferences and sending out notifications of that type.

There are two main categories of notification types: subscription and push. These behave differently. Refer to [Push vs. Subscription Notification Types](#) for more information.

Push notification types are configured as either parent or individual push notification types. A parent push notification type is used to define communication preferences and an individual push notification type is used to send notifications. Refer to [Parent vs. Individual Push Notification Types](#) for more information.

The base product supplies one Notification Type business objects. For more details, refer to the specifications for this business object.

Some notification types require specific configuration. Refer to [Designing Notification Types](#) for more information.

To set up notification types, select **Admin menu > Notification Type +**.

The [Notification Type Portal](#) contains the following zones:

- [Notification Type List](#)
- [Notification Type Zone](#)

Notification Type Portal

This section describes the zones associated to the Notification Type Portal:

- [Notification Type List](#)
- [Notification Type Zone](#)

Notification Type List

The Notification Type List zone lists all notification types. The following functions are available:

Click a [broadcast](#) button to open other zones that contain more information about the adjacent notification type.

The standard actions of Edit, Delete, and Duplicate are available for each notification type.

Click the Add link in the zone's title bar to add a new notification type.

Notification Type Zone

The Notification Type zone contains display-only information about the Notification Type. This zone appears when a Notification Type has been broadcast from the Notification Type List zone or if this portal is opened via drill down from another page.

Please see the zone's help text for information about this zone's fields.

Where Used

Follow this link to open the data dictionary where you can view tables that reference [CI_NTF_TYPE](#).

Setting Up the Outbound Message Type

An outbound message type is required for the notification outbound message. This outbound message type must reference the base [CI-NotifPrefOutboundMessage](#) (Notification Preference Outbound Message) business object.

Setting Up the Message Sender

A Message Sender is required to define how to send notifications. Use the context of the Message Sender to define the web service interface.

Setting Up the External System and Configure the Messages

Define an external system and configure the valid outbound message types and the method of communication for each (XAI, Batch, or Real Time); Real Time is generally the appropriate choice notifications).

Setting Up Notification Preferences Master Configuration

General configuration details for the notification preferences functionality is captured in the **Notification Preferences**[master configuration](#).

For more information about specific fields in the master configuration, refer to the inline help.

Other Edge Application Notification

Other applications such as *Oracle Utilities Network Management System* or *Oracle Utilities Meter Data Management* may initiate notifications; for example, the network management product can create notifications for outages or restoration information. The customer's preferences to receive these notifications will reside in Oracle Utilities Customer Care and Billing, since this is where their contact information exists. The other applications need to be notified when an active contact preference is created for one of their notifications. The **Notification Preferences**[master configuration](#) provides the section to indicate if a Notification Type is owned by another edge application. When a contact preference is created or inactivated for one of these notification types, an outbound message is sent to notify the other edge product. When the system event in another edge application occurs that initiates a notification, that application calls and inbound web service that creates the notification.

The **Notification Preferences**[master configuration](#) help text contains detailed information about other edge applications.

Setting Up Service Agreement Options

This section describes tables that must be set up before you can define service agreements.

Setting Up Standard Industry Codes (SIC)

A service agreement for non-residential service should reference a standard industry code (SIC). This code is used to categorize service agreements for reporting purposes. To define a SIC, open **Admin > Customer > SIC Code**.

Description of Page

Enter a unique **SIC Code** and **Description** for the SIC.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_SIC](#).

Setting Up Tax Exempt Types

Your rates will probably have provisions for the calculation of taxes of one type or another. Frequently you will have customers who are completely or partially exempt from these taxes. The service agreements for these customers will need to have tax exemption information in order for them to be billed properly. Tax Exempt Type is used to define the precise nature of the applicable exemption. To define the Tax Exempt Types you will use, open **Admin > Tax Exempt Type**.

Description of Page

Enter a unique **Tax Exempt Type** and **Description** for each type of tax exemption.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_TAX_EX_TYPE](#).

Setting Up Contract Quantity Types

You may have customers whose contracts (service agreements) have contractual consumption limits. The service agreements for these customers must have information regarding this quantity in order to be billed properly. Contract Quantity Type is used to precisely define the nature of the quantity. To define the Contract Quantity Types, open **Admin > Rates > Contract Quantity Type**.

Description of Page

Enter a unique **Contract Quantity Type** and **Description** for each type of contract quantity.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_CONT_QTY_TYP](#).

SA Type Controls Everything

Every service agreement references a SA type. The SA type controls all aspects of a service agreement's behavior including how service is started, how bills are created, how its financial transactions are booked in the general ledger, and much more. We don't explain how to set up SA types in this section because it's only after you have set up all of the control tables in this manual that you'll be able to finally define your SA types.

FASTPATH:

For more information about SA types, refer to [Defining Service Agreement Types](#).

Financial Controls

FASTPATH:

There are also a number of control tables that must be set up to control the bills, payments, and adjustments that are linked to a service agreement. For more information about these tables, please refer to [Defining Financial Transaction Options](#).

Setting Up Order Options

This section describes tables that must be set up before orders can be used to start service.

FASTPATH:

For more information, refer to [The Big Picture of Campaigns, Packages and Orders](#).

Setting Up Column References

A column reference must be created for each miscellaneous field that's captured on an order that doesn't reside in a characteristic. Refer to [Determine The Properties Of Every Miscellaneous Field](#) for more information.

Open **Admin > Sales & Marketing > Column Reference > Add** to define your column references.

Description of Page

Enter an easily recognizable **Column Reference** code and **Description** for each column reference.

Specify the **FK Reference** to use if this column reference uses field values from another table. Use **Long Description** to describe the data that fields using this column reference capture.

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated. You can define algorithms for the following system events: Post when order completed, Retrieve current value, Validate field value, Service task order processing, and Pre-process field value. Refer to [Extract Column References](#) for a description of these events.
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

WARNING:

These algorithms are typically significant processes. The absence of an algorithm may prevent the system from operating correctly.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COL_REF](#).

Setting Up Order Cancellation Reasons

An order cancellation reason must be supplied when an order is cancelled. Open **Admin > Order Cancel Reason** to define your reason codes.

Description of Page

Enter an easily recognizable **Order Cancel Reason** and **Description** for each order cancellation reason.

Where Used

Cancellation reasons are used when an [order is canceled](#).

Setting Up Order Hold Reasons

An order hold reason must be supplied when an order is held. Open **Admin > Sales & Marketing > Order Hold Reason** to define your reason codes.

Description of Page

Enter an easily recognizable **Order Hold Reason** and **Description** for each order hold reason.

Where Used

Hold reasons are used when an [order is held](#).

Setting Up Order Feature Configurations

Defining a [feature configuration](#) with a feature type of Order Configuration can increase performance of the Order page when campaigns have a large number of packages or criteria. Open **Admin > Feature Configuration > Add** to define a configuration for the feature type Order Configuration.

NOTE:

Only one. The system expects only one order configuration feature configuration to be defined.

Description of Page

The following points describe the various **Option Types** that may be defined:

- Eligibility Tree - Suppress Error Packages node. Select this option type and define a value if you would like the [Order Eligibility Tree](#) to suppress the node that contains packages with errors in their eligibility criteria. This is an optional setting. If the option type is not defined, the error packages node is displayed, if applicable.
- Eligibility Tree - Suppress Ineligible Packages node. Select this option type and define a value if you would like the [Order Eligibility Tree](#) to suppress the node that contains packages that are not applicable to the customer based on the eligibility criteria. This is an optional setting. If the option type is not defined, the ineligible packages node is displayed, if applicable.
- Eligibility Tree - Suppress Other Campaigns node. Select this option type and define a value if you would like the [Order Eligibility Tree](#) to suppress the node that contains other eligible campaigns. This is an optional setting. If the option type is not defined, the other campaigns node is displayed, if applicable.

Setting Up Program Management Options

This section describes how to set up the system for program management.

For more information about program management, see [The Big Picture of Program Management](#)

Designing Initiative Eligibility Criteria

The following describes guidelines for designing initiative eligibility criteria. These guidelines apply when using the base-supplied business objects and any additional business objects that the implementation defines.

There are two general types of initiative eligibility criteria business objects provided in base: specific and freeform. Your implementation decides whether to use only one of these types or a mixture of both.

NOTE: Initiative criteria has three categories: eligibility criteria, related object criteria, and participation criteria. (The participation criteria type is currently not used.) During criteria maintenance and lead generation, the system identifies the category to use through a business object option called Initiative Criteria Type. Eligibility criteria business objects are defined with an Initiative Criteria Type of either C1EC (Eligibility Criteria) or C1BT (Both Eligibility Criteria and Participation Criteria).

Specific Criteria Business Objects

Most of the base-supplied business objects are designed to perform comparisons to a specific field. The algorithm type for retrieving/determining the value to compare with is already preset in the business object, such that the Algorithm values that the user sees on the user interface are restricted to the instances of that algorithm type.

Refer to the following base business objects for more details:

- Communication Channel Criteria
- Conservation Program Criteria
- Contract Option Criteria
- Rate Schedule Criteria
- SP Type Criteria
- Service Type Criteria
- Is Account Current Criteria
- Account is on Autopay Criteria
- Account is on Budget Criteria
- Number of Days Since Last Lead Criteria
- Account Customer Class Criteria
- Average Service Quantity Criteria
- Premise Cities Criteria

To add similar business objects:

- Identify the specific value for comparison. This could be a reference to an object or entity in the system, or a character string; such as, true/false.
- Develop the algorithm program to retrieve or determine the comparison value.
- Configure your algorithm type using the Initiative Criteria - Derive Field Value algorithm entity and reference the program you created.

- Configure your business object schema so that the Algorithm field retrieves all algorithm instances of the algorithm type you created.
- Define algorithm instances of your algorithm type.

Freeform Criteria Business Object

The Freeform Criteria business object, on the other hand, is designed to perform comparisons for different types of values. The comparison value is determined by a Derive Value Algorithm Type that is defined as an option on this business object. A number of these Derive Value Algorithm Types are supplied with this business object and your implementation can add to these, as necessary.

NOTE: The supplied Derive Value Algorithm Types perform similar comparisons as those in the base-supplied specific criteria business objects (listed above). The Freeform Criteria business object should only be used if the available specific criteria business objects are not sufficient to perform the needed field value comparisons.

To add Derive Value Algorithm Types:

- Identify the specific value for comparison. It could be a reference to an object or entity in the system, a character string (such as, true/false), or a scalar value (such as, number of days or number of kwh).
- Develop the algorithm program to retrieve or determine the comparison value.
- Configure your algorithm type using the Initiative Criteria - Derive Field Value algorithm entity and reference the program you created.
- Define an algorithm instance for your algorithm type.
- Update the Freeform Criteria business object to add a Derive Value Algorithm Type business object option referencing your algorithm.

Repeat the above steps for each specific field to compare.

Designing Related Object Criteria

The following describes guidelines for designing related object criteria. These guidelines apply when using the base-supplied business objects and additional business objects that the implementation defines.

An initiative defines the related object(s) to associate with leads via the Object Nomination algorithm on the initiative business object. The lead generation process executes this object nomination algorithm after executing initiative eligibility criteria. The base supplied parent Initiative business object (C1–Initiative) is configured with an object nomination algorithm that retrieves the premises that are associated with the account’s active and pending start service agreement’s service points. This algorithm applies the related object criteria, which is configured on the initiative, to each premise. If the premise is determined to be eligible, it is returned as a related object for the lead. To override the base object nomination algorithm, add another instance of the Object Nomination System Event and specify your own algorithm.

The base product supplies related object criteria business objects that look for one or more premise to associate with a lead.

There are two general types of related object criteria business objects provided in base - specific and freeform. Your implementation decides whether to use only one of these types or a mixture of both.

NOTE: Initiative criteria has three categories: eligibility criteria, related object criteria, and participation criteria. (The participation criteria type is currently not used.) During criteria maintenance and lead generation, the system identifies the category to use through a business object option called Initiative Criteria Type. Related object criteria business objects are defined with an Initiative Criteria Type of C1EO (Related Object Criteria).

Specific Criteria Business Objects

The base product supplies two business objects that perform comparisons to a specific premise-related field. The algorithm type for retrieving or determining the comparison value is preset in the business object, such that the Algorithm values that the user sees on the user interface are restricted to the instances of that algorithm type.

Refer to the following base business objects for more details:

- Premise SP Type Criteria
- Geographic Value Criteria

To add similar business objects:

- Identify the specific comparison value. It could be a reference to an object or entity in the system (such as, SP Type), or any other attribute that can be used to find the related object.
- Develop the algorithm program to retrieve or determine the comparison value.
- Configure your algorithm type using the Initiative Criteria - Derive Field Value algorithm entity and reference the program you created.
- Configure your business object schema so that the Algorithm field retrieves all algorithm instances of the algorithm type you created.
- Define algorithm instances of your algorithm type.

Freeform Premise Criteria Business Object

The Freeform Premise Criteria business object, on the other hand, is designed to perform comparisons to different types of values. The comparison value is determined by a Derive Value Algorithm Type that is defined as an option on this business object. Derive Value Algorithm Types are supplied with this business object and your implementation can add to these, as necessary.

NOTE: The supplied Derive Value Algorithm Types perform similar comparisons as those in the base-supplied specific criteria business objects (listed above). The Freeform Premise Criteria business object should only be used if the available specific criteria business objects are not sufficient to perform the needed field value comparisons.

To add Derive Value Algorithm Types:

- Identify the specific comparison value.
- Develop the algorithm program to retrieve or determine the comparison value.
- Configure your algorithm type using the Initiative Criteria - Derive Field Value algorithm entity and reference the program you created.
- Define an algorithm instance for your algorithm type.
- Update the Freeform Premise Criteria business object to add a Derive Value Algorithm Type business object option referencing your algorithm.

Repeat the above steps for each specific field to compare.

Setting Up Lead Event Types

Each initiative can define a structured marketing effort for the events that take place during the life of a lead. For example, one event could be that a letter or an email is sent to the customer.

Each event references a lead event type. An initiative defines a lead event type profile that specifies one or more lead event types and the sequence in which those event types should occur. Refer to [The Big Picture of Initiatives](#) for more information.

Some types of lead events may also involve sending outbound messages to an external system — e.g., to notify a third party representative about a customer who has signed up. The lead event type business object can specify the outbound message type(s) to use.

Event types that involve additional workflow activities can define one or more steps; for instance, when work is outsourced to a third party representative (after customer sign-up), the third party lead event includes a list of steps to complete. Each event step has information (such as, completion date) that can be used to track progress.

The base product supplies a number of Lead Event Type business objects. For more details, refer to the specifications for these business objects.

To set up lead event types, select **Admin > Lead Event Type**.

The Lead Event Type portal contains the following zones:

Lead Event Type List

The Lead Event Type List zone lists all lead event types. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent lead event type.
- The standard actions of **Edit**, **Delete** and **Duplicate** are available for each lead event type.

Click the **Add** link in the zone's title bar to add a new lead event type.

Lead Event Type

The Lead Event Type zone contains display-only information about the Lead Event Type. This zone appears when a Lead Event Type has been broadcast from the Lead Event Type List zone or if this portal is opened via drill down from another page.

Please see the zone's help text for information about this zone's fields.

Where Used

Follow this link to open the data dictionary where you can view tables that reference [C1_LEAD_EVT_TYPE](#).

Overriding Initiative Sign Up Options

The behavior of the Sign Up action on Lead maintenance is controlled by two business object options on the initiative:

- Sign Up UI Map
- Sign Up Service Script

When invoking a lead's Sign Up action, the system retrieves these options and displays or executes them accordingly.

The base-owned parent initiative business object, C1-Initiative, is supplied with the Sign Up Service Script option referencing the C1-IntvSignU script. To override this, add another *Sign Up Service Script* option with a higher sequence number.

The Sign Up UI map, on the other hand, is not plugged in on the base BO, because this could vary across implementations. The Sign Up UI map C1-SignUpResponseCh provided in base simply prompts for a Response Channel. Implementations can use this map or their own specific map.

NOTE: The system does not issue an error if the Sign Up UI map is not specified. It proceeds with executing the Sign Up Service Script and, if there are no problems, displays a sign up confirmation. The base-supplied C1-IntvSignU script includes logic to recheck the account's eligibility for the initiative and update the *Has Signed Up* indicator on the lead.

Program Management Master Configuration

The master configuration business object for Program Management defines general configuration details for the program management functionality.

Navigate using **Admin > General > Master Configuration**. In the Master Configuration list, scroll to Program Management Configuration and click the add icon if there is no record yet or the edit icon to modify an existing record.

For more information about specific fields in the master configuration, refer to the embedded help.

Defining Field Order Options

A field order is a group of field activities that take place at a premise's service point(s). These activities can range from the simple (e.g., read a meter) to the complex (e.g., install both the power line and a new meter). Before you can issue field orders, you must establish the control data defined in this section.

NOTE:

Appointments. Refer to [The Big Picture of Appointments](#) for information about how appointments can be scheduled for field activities.

WARNING:

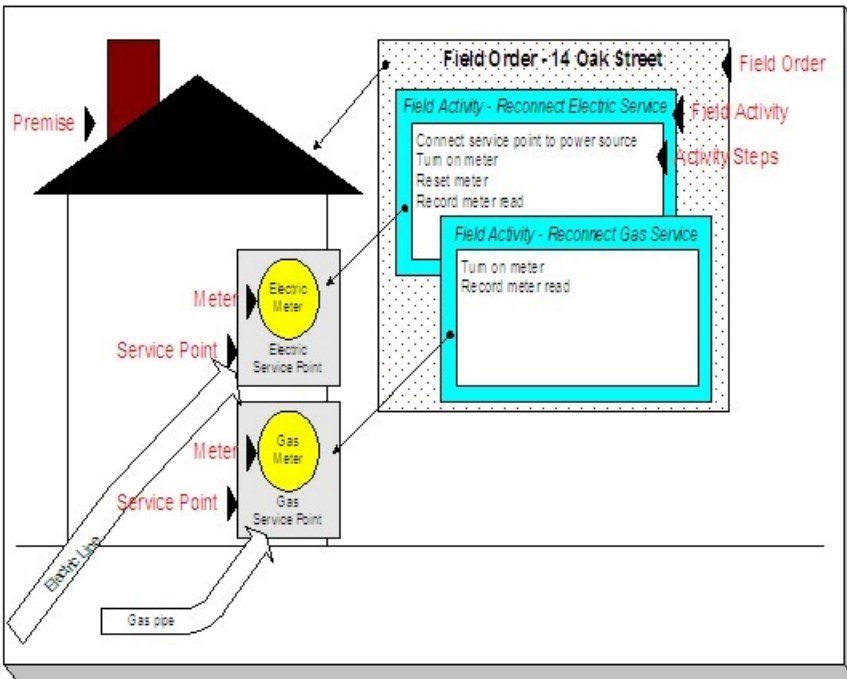
Setting up the tables that control your field activities will be as complicated as the fieldwork your organization performs. If your company doesn't do fieldwork, then you won't have to set up any of these tables. If your company has a single service and the fieldwork you perform is straightforward, this setup process will be straightforward. If your company performs sophisticated fieldwork (e.g., utilizing multiple crews and multiple dispatch locations), this setup process will require careful analysis.

FASTPATH:

For more information about field orders and how they use the information described in this chapter, refer to [The Big Picture Of Field Orders](#).

An Example Of A Field Order

The following picture illustrates a field order that controls the work to be performed at a premise with 2 service points.



The following field order-related concepts are illustrated above:

Field Order A field order is a group of field activities performed by one person (or crew) at a premise. Refer to [How Are Field Orders Created And Dispatched](#) for information about how field orders are created.

Field Activity A field activity is a task that takes place at a service point. Examples of field activities include reconnect service, exchange meter, disconnect service, cut for nonpayment, investigate trouble order.

The system automatically creates field activities when specific events happen. Refer to [Designing Your Field Activity Profiles](#) for a discussion of how the system does this.

When a field activity is first added, its state is pending. If the activity is not done (for whatever reason), the activity is canceled. After the activity is done, the results of the activity are recorded in the system and its state becomes complete.

Activity Step A field activity has one or more steps. For example, the field activity to exchange a meter would have the following activity steps: Remove existing meter, Test meter, Install new meter.

The number and types of steps involved with a field activity are controlled by the activity's activity type. Refer to [Setting Up Field Activity Types](#) for more information.

Premise See [An Illustration Of A Premise](#) for a description.

Service Point See [An Illustration Of A Premise](#) for a description.

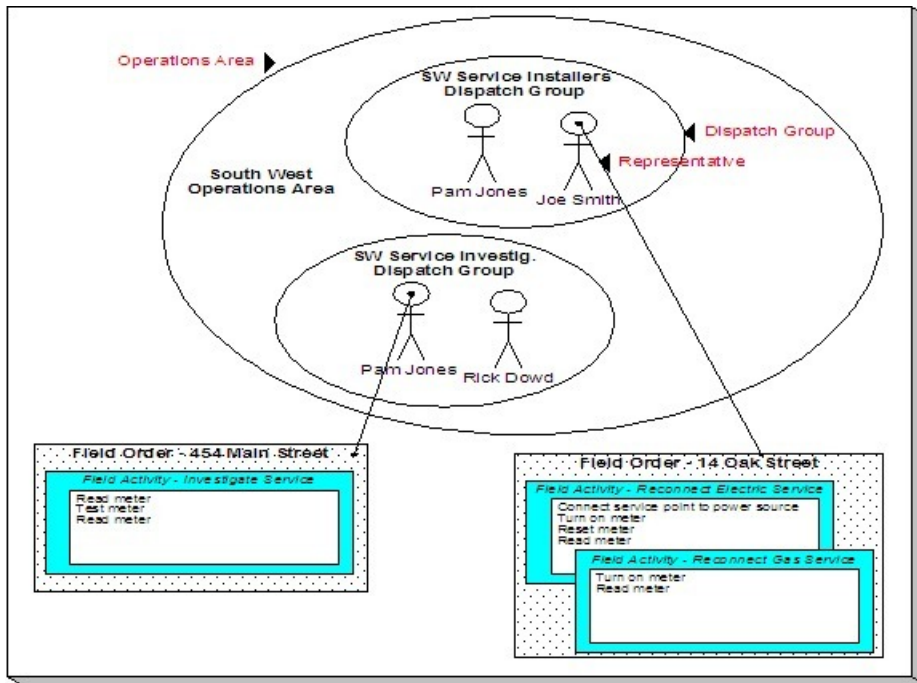
Meter See [An Illustration Of A Premise](#) for a description.

NOTE:

More than just meters. While the above diagram illustrates field order concepts in respect of meters, the field orders system has been designed to handle meters, lamps, and any other type of device located at a service point (e.g., a cable box).

An Example Of The Entities Involved In Field Order Dispatch

The following picture illustrates two field orders that are dispatched from the same operations area where each field order is assigned to a different representative.



The following dispatch-related concepts are illustrated above:

Representative A representative is the individual (or crew) that performs a field order's activities. A representative may work in any number of dispatch groups. Refer to [Setting Up Representatives](#) for more information.

Operations Area An operation area is a physical or logical location from which field orders are dispatched. Every service point references the operation area(s) responsible for its service. Refer to [Setting Up Operations Areas](#) for more information.

Dispatch Group A dispatch group is a logical group of representatives located at an operations area. For example, in the South West Operations Area you may have the dispatch groups of SW Service Installers, SW Service Investigators, SW Meter Exchangers, etc. Within each dispatch group are representatives with interchangeable skills (i.e., you can assign a field activity to any representative within a dispatch group). Refer to [Setting Up Dispatch Groups](#) for more information.

The system automatically assigns a field activity a dispatch group when it's first created. It does this based on: 1) the type of field activity, 2) the service point's SP type, and 3) the operation area on the service point that's linked to the field activity type's field service classification. Refer to [Designing Who Does Your Field Activities](#) for more information.

Setting Up Representatives

A representative is the individual (or equipment) that performs a field order's activities. At dispatch time, a representative may be assigned to a field order. To define your organization's representatives, open **Admin > Field Work > Representative**.

Description of Page

Enter an easily recognizable **Representative** and **Description** for each representative.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_REP](#).

Setting Up Operations Areas

When you set up a service point, you must define the operation areas that manage its fieldwork.

NOTE:

How a service point gets its operation area(s). A service point's operation areas default based on its service type and its premise's postal code. See [Setting Up Premise & Service Point Postal Defaults](#) for more information.

To define your organization's operation areas, open **Admin > Operations Area**.

NOTE:

A service point may have multiple operation areas. For example, a service point may have one operations area for installs and removals, a separate operation area for trouble orders, and yet a third operation area for meter exchanges. Refer to [Setting Up Field Service Classifications](#) for more information about how different types of fieldwork can be classified.

Description of Page

Enter an easily recognizable **Operations Area** and **Description** for each operation area.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_OP_AREA](#).

Setting Up Dispatch Groups

A dispatch group is a logical group of representatives located at an operations area. When a field activity is created, the system assigns it to a dispatch group based on the type of activity, the type of service point, and the operations area that manages the service point. The topics in this section describe the pages used to maintain a dispatch group.

Dispatch Group - Main

To define your organization's dispatch groups, open **Admin > Field Work > Dispatch Group > Add**.

Description of Page

Enter an easily recognizable **Dispatch Group ID** and **Description** for each dispatch group.

Turn on **Allow Dispatch** if orders allocated to this dispatch group are "dispatchable".

NOTE:

Trouble orders without a premise. The only example we can think of where Allow Dispatch is off would be for the dispatch group associated with trouble orders without a premise. These trouble orders are associated with a "dummy" service point, and the dummy service point's operations area should reference the unknown dispatch group. Someone would need to periodically look at the unknown dispatch group and associate its field activities with the appropriate service point and dispatch group.

If this dispatch group is "dispatchable" and work for this dispatch group is interfaced to an external system, indicate the appropriate external system [feature configuration](#) or outage management feature configuration for the dispatch group.

NOTE:

Separate module. Functionality related to integrating with an [external system](#) or an outage system is associated with separate function modules. The feature configuration field is not available if these modules are [turned off](#).

Batch Control along with the Field Order Extract algorithm defined on the **Algorithms** tab control how field orders that reference this dispatch group are printed (both in batch and online). Refer to [Technical Implementation Of Batch Field Order Production](#) for more information about producing field orders in batch. Refer to [Technical Implementation Of Online Field Order Production](#) for more information about online field order production.

- Use **Batch Control** to define the process that creates the flat file that is passed to your field order printing software. If you use an **Extract Algorithm** to construct the downloaded information, you can use either the [FODL](#) or the [DSGPFODL](#).
- Use the Field Order Extract algorithm defined on the **Algorithms** tab to define the plug-in component that constructs the "flat file records" that contain the information to be merged onto field orders that reference this dispatch group. This algorithm is called when a user requests an online image of a field order on [Field Order - Main](#) and it may also be called by the batch field order extraction process defined above. Please be aware that the system comes with a sample algorithm type - [FOEX-OX](#) - that should be used as a sample if you have to write a new plug-in [algorithm](#).

Use **Alternative Dispatch Group** to define the dispatch group that can do everything that this dispatch group can do. This field is used when the system attempts to find a common dispatch group for all field activities associated with a premise (when such field activities don't have the same dispatch group) during the appointment scheduling process. Refer to [Using Alternate Dispatch Groups To Find The Lowest Common Denominator](#) for more information.

The **Representative** collection shows the representatives who may be assigned to field orders performed by a dispatch group.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DISP_GRP](#).

Dispatch Group - Field Activity Type Review

Open **Admin > Field Work > Dispatch Group > Search** and navigate to the **FA Type Review** page to review the field activities that can be performed by the dispatch group. For each field activity in the tree, you can view the SP types associated with the activity.

NOTE:

Four dimensions. For every **field activity type**, you define the **dispatch group** that performs the activity at every **SP type** located in every **operations area**. This information is maintained on the Field Activity Type page. This is a rather complex relationship because it involves the four dimensions highlighted in bold. Due to this complexity, we have provided review trees on the SP Type, Dispatch Group, and Field Activity Type pages to help you understand what you've set up.

Description of Page

This page is dedicated to a tree that shows the field activity types performed by the dispatch group. And for each field activity and operations area, you can view the applicable SP types.

Dispatch Group - Algorithm

Open **Admin > Field Work > Dispatch Group > Search** and navigate to the **Algorithm** page to define the algorithms that should be executed for field activities / field orders associated with a given dispatch group.

Description of Page

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (descriptions of all possible events are provided below).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

WARNING:

These algorithms are typically significant processes. The absence of an algorithm may prevent the system from operating correctly.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Available Appointments	Required if you use base appointment scheduling functions (and not external system integration)	<p>The algorithm plugged into this spot is responsible for determining which appointment periods may be linked to a field activity.</p> <p>Refer to Appointment Maintenance for a description of the transaction that's used to setup an appointment.</p> <p>Refer to The Big Picture Of Appointments for general information about appointments.</p> <p>Click here to see the algorithm types available for this system event.</p>
FA Integration	Required if dispatch group interfaces with an external system.	<p>The algorithm plugged into this spot is responsible for creating appropriate notification download staging records used to interface field activity information to an external system.</p> <p>This algorithm is not allowed if the dispatch group does not interface with an external system.</p> <p>Refer to Algorithms Control FA Integration for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Field Order Extract	Required if dispatch group is "dispatchable" (refer to the description of Allow Dispatch on the Main page for more information)	<p>Refer to the description of Batch Control on the Main tab for a description of this system event.</p> <p>Click here to see the algorithm types available for this system event.</p>
Validate Appointments	Optional if you use base appointment scheduling functions	<p>The algorithm plugged into this spot is responsible for determining if a field activity can be linked to an appointment period. If you don't plug-in a Validate Appointments algorithm, the system will allow any appointment period that's displayed on the Appointment Maintenance transaction to be linked to a field activity.</p>

Refer to [Appointment Maintenance](#) for a description of the transaction that's used to setup an appointment.

Refer to [The Big Picture Of Appointments](#) for general information about appointments.

Click [here](#) to see the algorithm types available for this system event.

Defining Disconnect Locations

When a service point is disconnected from the supply source, a disconnect location must be specified. This location defines where service was severed. It also controls the type of field activity generated to reconnect service.

NOTE:

A service point's disconnect location is updated as part of the service disconnection process. This location is used to determine the appropriate crew to send out when it's time to reconnect service.

To define disconnect location codes, open **Disconnect Location** page.

Description of Page

Enter a **Disconnect Location** and **Description** for every disconnect location.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_DISCON_LOC](#).

Designing Your Field Activity Profiles and Types

A field activity is a task that takes place at a service point. Every field activity references a field activity type. The field activity type defines the steps involved in the execution of the activity.

Most field activities are created:

- When a customer service representative starts or stops service at a premise, the system automatically creates field activities to perform the necessary fieldwork based on the type and state of the service point.
- When service is cut due to lack of payment.

If you set the system up correctly, your CSR's won't have to create these field activities. Rather, the system creates field activities based on the information you set up in your field activity type profiles. You typically have a different field activity profile for every major category of service point. We recommend that you familiarize yourself with the following documentation before you set up your field activity type profiles:

- Refer to [Starting Service & Field Activities](#) for a description of how field activities are created to start service.
- Refer to [Stopping Service & Field Activities](#) for a description of how field activities are created to stop service.
- Field activities used to cut service due to non-payment are created by two modules:
 - If you use severance process to stop a service agreement due to nonpayment, refer to [Field Events And Their Activities](#) for a description of how these field activities are created.
 - If you use cut processes to stop a service agreement due to nonpayment, refer to [Field Activities To Cut and Reconnect Service](#) for a description of how these field activities are created.

The topics in this section describe how to design your field activity profiles and field activity types.

How Does A Field Activity Type Profile Get Used?

A field activity type profile contains:

- A list of field activity types that can be performed at service points.
- Matrices defining the specific activity type(s) to generate in order to start service, stop service, or disconnect due to non-payment.

You may wonder how field activity type profiles get related to your service points. It's a little indirect, but the indirection provides a great deal of flexibility:

- Every service point references an SP type.
- Every SP type references the specific field activity type profile used by the start/stop and credit and collections process to generate field activities.

An example will help illustrate how this works:

- When you start service at a specific service point, the system extracts the service point's SP type.
- Then, the system determines the field activity type profile that is to be used on this SP type.
- Then, the system determines the state of the service point (e.g., connected, no meter).
- And finally, it generates the appropriate field activity.

Designing Field Activity Type Profiles

The number of field activity type profiles is dependent on a variety of factors:

- If your field crews (i.e., dispatch groups) are highly specialized, you'll need more profiles than when your crews can perform a variety of activities. For example, if you have electric installers who just install electric service and gas installers who just install gas service, you'll need a different profile for electricity versus gas.
- If the field activities you perform differ based on the type of service point, you'll need more profiles than if you have standard activities. For example, if you turn off residential service differently than you turn off commercial service, you'll need a profile for each type of customer.
- If you have meters and item-based service, you'll need a separate profile to distinguish the field activities for these two types of service.
- If more than one of the above points applies to your organization, you'll need a separate profile for the superset.

The easiest way to design your profiles is to pick an SP type and design its profile by filling in matrices similar to those defined below (choose either meter or item-based as per the SP type). After you've designed the profile, determine how many other SP types on which it can be used. Then design the next SP type's profile and determine where it can be reused. Repeat this process until all your SP types have a profile.

FASTPATH:

In order to design your field activity type matrices, you must have a good understanding of field activity types and how they control the actions performed by your field crews. Refer to [Setting Up Field Activity Types](#) for more information.

Typical Metered Service Matrix

In this section, we provide a sample of how the activity type matrix would look for metered service. The topics show two matrices; one for start/stop field activities, the other for credit & collections activities. In reality, there is a separate matrix for every column in the matrices.

Starting and Stopping Metered Service Matrix

The following matrix is representative of how a metered SP type's field activity type profile would look. The following points describe the rows, columns and cells in the matrix:

- The customer events represent the three start and stop events that may take place at the service point.
 - Start service is the event that takes place when a service point is to be started and the system has no knowledge of a related stop event.
 - Stop service is the event that takes place when a service point is to be stopped and the system has no knowledge of a related start event.
 - Start/stop service is the event that takes place when a service point is to be stopped for one customer and started for another at the same time.
- The SP conditions represent the state of service at a service point. The following conditions are represented:
 - A device (the meter) is either installed or not installed.
 - The installed device is either turned on or off.
 - The source of service (the line or pipe) is either connected to the service point or disconnected.
 - If the source is disconnected, define a field activity for each potential disconnect location.
- Each cell in the matrix represents a field activity to be created by the system for the combination of customer event and service point condition.

SP Condition	Customer Event:	Customer Event:	Customer Event:
	Start Service	Stop Service	Start/Stop Service
No Device / SP Disconnected at meter	Install meter and connect line	N/A - no field work is necessary because the meter should have been read when it was removed	N/A
No Device / SP Disconnected at pole	Install meter and connect line	N/A - no field work is necessary because the meter should have been read when it was removed	N/A
No Device / SP Connected	Install meter	N/A - no field work is necessary because the meter should have been read when it was removed	N/A
Device Installed / Device Off / SP Disconnected at meter	Connect line and turn on meter	N/A - no field work is necessary because the meter should have been read when the service was severed	N/A
Device Installed / Device Off / SP Disconnected at pole	Connect line and turn on meter	N/A - no field work is necessary because the meter should have been read when the service was severed	N/A
Device Installed / Device Off / SP Connected	Turn on meter	N/A - no field work is necessary because the meter should have been read when the meter was turned off	N/A

Device Installed / Device On / SP Connected	Read meter	Turn off meter	Read meter
Device Installed / Device On / SP Disconnected at meter	Connect line and read meter	N/A - no field work is necessary because the meter should have been read when the line was disconnected	N/A
Device Installed / Device On / SP Disconnected at pole	Connect line and read meter	N/A - no field work is necessary because the meter should have been read when the line was disconnected	N/A

Cutting Metered Service Due To Nonpayment

The following matrix is representative of how a metered SP type's field activity type profile might look. The following points describe the rows, columns and cells in the matrix:

- The customer events represent the three C&C-oriented field activities that may take place at the service point.
 - Disconnect warning is the event that warns the customer of imminent severance if payment is not received.
 - Cut for non-payment is the event that causes service to be severed due to non-payment.
 - Reconnect is the event that takes place when a service point is reconnected because payment was received after service was cut.
- The SP conditions represent the state of service at a service point. The following conditions are represented:
 - A device (the meter) is either installed or not installed.
 - The installed device is either turned on or off.
 - The source of service (the line or pipe) is either connected to the service point or disconnected.
 - If the source is disconnected, define a field activity for each potential disconnect location.
- Each cell in the matrix represents a field activity to be created by the system for the combination of customer event and service point condition.

SP Condition	Customer Event:	Customer Event:	Customer Event:
	Disconnect Warning	Cut For Non-Payment	Reconnect
Device Installed / Device On / SP Connected	Leave disconnection warning	Cut metered serviced	N/A
Device Installed / Device Off / SP Connected	N/A - service is off	N/A - service is off	Turn on meter
Device Installed / Device On / SP Disconnected at pole	N/A - service is off	N/A - service is off	Connect line
Device Installed / Device Off / SP Disconnected at pole	N/A - service is off	N/A - service is off	Connect line and turn on meter
No Device / SP Connected	N/A - service is off	N/A - service is off	Install meter
No Device / SP Disconnected at pole	N/A - service is off	N/A - service is off	Install meter and connect line

Meter Testing Activities

While the field activity types associated with meter testing should not appear in the above matrixes, these field activity types must be defined in the metered service points' field activity type profiles (under Valid Activity Types). Refer to [Examples of Device Testing Activity Types and their Steps](#) for example of these field activity types.

Typical Badged Item Service Matrix

In this section, we provide a sample of how the activity type matrix would look for BADGED item-based service (e.g., badged lamps). The topics show two matrices; one for start/stop field activities, the other for credit & collections activities. In reality, there is a separate matrix for every column in the matrices.

FASTPATH:

Refer to [Service Points \(SPs\)](#) for more information about the difference between badged and non-badged items.

Starting and Stopping Badged Item Service Matrix

The following matrix is representative of how a lamp SP type's field activity type profile would look. The following points describe the rows, columns and cells in the matrix:

- The customer events represent the three start and stop events that may take place at the service point.
 - Start service is the event that takes place when a service point is to be started and the system has no knowledge of a related stop event.
 - Stop service is the event that takes place when a service point is to be stopped and the system has no knowledge of a related start event.
 - Start/stop service is the event that takes place when a service point is to be stopped for one customer and started for another at the same time.
- The SP conditions represent the state of service at a service point. The following conditions are represented:
 - A device (the item) is either installed or not installed.
 - The installed device is either turned on or off.
 - The source of service (the line or pipe) is either connected to the service point or disconnected.
 - If the source is disconnected, define a field activity for each potential disconnect location.
- Each cell in the matrix represents a field activity to be created by the system for the combination of customer event and service point condition.

SP Condition	Customer Event:	Customer Event:	Customer Event:
	Start Service	Stop Service	Start/Stop Service
No Device / SP Disconnected at pole	Install lamp and connect line	N/A - no field work is necessary because there is no lamp	N/A
No Device / SP Connected	Install lamp	N/A - no field work is necessary because there is no lamp	N/A
Device Installed / Device Off / SP Disconnected at pole	Connect line and install eye	N/A - no field work is necessary because the lamp is off	N/A

Device Installed / Device Off / SP Connected	Install eye	N/A - no field work is necessary because the lamp is off	N/A
Device Installed / Device On / SP Disconnected at pole	Connect line	N/A - no field work is necessary because the lamp is off	N/A
Device Installed / Device On / SP Connected	No field work necessary	Turn off lamp	No field work necessary

Cutting Badged Item Service Due To Nonpayment

The following matrix is representative of how an item-based SP type's field activity type profile would look. The following points describe the rows, columns and cells in the matrix:

- The customer events represent the three C&C-oriented field activities that may take place at the service point.
 - Disconnect warning is the event that warns the customer of imminent severance if payment is not received.
 - Cut for non-payment is the event that causes service to be severed due to non-payment.
 - Reconnect is the event that takes place when a service point is reconnected because payment was received after service was cut.
- The SP conditions represent the state of service at a service point. The following conditions are represented:
 - A device (the item) is either installed or not installed.
 - The installed device is either turned on or off.
 - The source of service (the line or pipe) is either connected to the service point or disconnected.
 - If the source is disconnected, define a field activity for each potential disconnect location.
- Each cell in the matrix represents a field activity to be created by the system for the combination of customer event and service point condition.

SP Condition	Customer Event:	Customer Event:	Customer Event:
	Disconnect Warning	Cut For Non-Payment	Reconnect
Device Installed / Device On / SP Connected	Leave disconnect warning	Cut lamp service	N/A
Device Installed / Device Off / SP Connected	N/A - service is off	N/A - service is off	Turn on item
Device Installed / Device On / SP Disconnected at pole	N/A - service is off	N/A - service is off	Connect line
Device Installed / Device Off / SP Disconnected at pole	N/A - service is off	N/A - service is off	Connect line and turn on item
No Device / SP Connected	N/A - service is off	N/A - service is off	Install item
No Device / SP Disconnected at pole	N/A - service is off	N/A - service is off	Install item and connect line

Item Testing Activities

While the field activity types associated with item testing should not appear in the above matrixes, these field activity types must be defined in the item-based service points' field activity type profiles (under Valid Activity Types).

Typical Unbadged Item Service Matrix

In this section, we provide a sample of how the activity type matrix would look for UNBADGED item-based service (e.g., parking lots, sewage service). The topics show two matrices; one for start/stop field activities, the other for credit & collections activities. In reality, there is a separate matrix for every column in the matrices.

Starting and Stopping Unbadged Item Service Matrix

The following matrix is representative of how a field activity type profile would look for a parking lot's SP type. The following points describe the rows, columns and cells in the matrix:

- The customer events represent the three start and stop events that may take place at the service point.
 - Start service is the event that takes place when a service point is to be started and the system has no knowledge of a related stop event.
 - Stop service is the event that takes place when a service point is to be stopped and the system has no knowledge of a related start event.
 - Start/stop service is the event that takes place when a service point is to be stopped for one customer and started for another at the same time.
- The SP conditions represent the state of service at a service point. The following conditions are represented:
 - The source of service (the line or pipe) is either connected to the service point or disconnected.
 - If the source is disconnected, define a field activity for each potential disconnect location.
 - Notice that the SP Conditions always indicate "No Device". This is because a device (i.e., a meter or badged item) cannot be installed at this type of service point.
- Each cell in the matrix represents a field activity to be created by the system for the combination of customer event and service point condition.

SP Condition	Customer Event:	Customer Event:	Customer Event:
	Start Service	Stop Service	Start/Stop Service
No Device / SP Disconnected at pole	Connect service in parking lot	N/A - no field work is necessary because there is no lamp	N/A
No Device / SP Connected	No field activity necessary	Disconnect service in parking lot	No field activity necessary

Cutting Unbadged Item Service Due To Nonpayment

The following matrix is representative of how an item-based SP type's field activity type profile would look. The following points describe the rows, columns and cells in the matrix:

- The customer events represent the three C&C-oriented field activities that may take place at the service point.
 - Disconnect warning is the event that warns the customer of imminent severance if payment is not received.
 - Cut for non-payment is the event that causes service to be severed due to non-payment.

- Reconnect is the event that takes place when a service point is reconnected because payment was received after service was cut.
- The SP conditions represent the state of service at a service point. The following conditions are represented:
 - Notice that the SP Conditions always indicate "No Device". This is because a device (i.e., a meter or badged item) cannot be installed at this type of service point.
 - The source of service (the line or pipe) is either connected to the service point or disconnected.
 - If the source is disconnected, define a field activity for each potential disconnect location.
- Each cell in the matrix represents a field activity to be created by the system for the combination of customer event and service point condition.

SP Condition	Customer Event: Disconnect Warning	Customer Event: Cut For Non-Payment	Customer Event: Reconnect
No Device / SP Connected	Parking lot disconnect warning	Cut parking lot service	Reconnect parking lot service
No Device / SP Disconnected at pole	N/A - service is off	N/A - service is off	Reconnect parking lot service

Designing Field Activity Types

The number of activity types you need is related to the different field activities your company performs at your service points. For example, if your company installs and maintains meters, you will set up field activity types for every conceivable meter-related task you assign to your field staff. If your company installs and maintains lamps, you will set up field activity types for every conceivable lamp-related task you assign to your field staff.

The topics in this section describe how to design your field activity types.

Designing Field Activity Types From Your Field Activity Type Profiles

After designing your field activity type profiles, the resulting matrices will reference every field activity type needed to:

- Start service
- Stop service
- Leave disconnect warnings
- Cut service due to non-payment
- Reconnect service after cut

The topics in this section illustrate every field activity type that would be needed to satisfy the needs of the field activity type profiles illustrated above.

Examples of Meter-Oriented Activity Types and their Steps

The following table shows several classic meter-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Connect SP and install meter	Connect SP to power source	Connect Service Point

	Install meter	Install Meter
	Attempt to notify owner - optional	Contact Customer
Connect SP	Connect SP to power source	Connect Service Point
Disconnect SP	Disconnect SP from power source	Disconnect Service Point
Install meter	Install meter	Install Meter
Remove meter	Remove meter	Remove Meter
Read meter	Read meter	Read Meter
Turn on meter	Turn meter on	Turn On Meter
Turn off meter	Turn meter off	Turn Off Meter
Reconnect meter after payment	Turn meter on - optional	Turn On Meter
	Install meter - optional	Install Meter
Disconnect warning	Place disconnect warning	Contact Customer
Cut meter for non payment	Turn meter off - optional	Turn Off Meter
	Remove meter - optional	Remove Meter
Remove and disconnect meter	Verify premise is vacant - optional	Contact Customer
	Remove meter	Remove Meter
	Disconnect SP from power source	Disconnect Service Point

Examples of Badged Lamp-Oriented Activity Types and their Steps

The following table shows several classic lamp-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Connect SP and install lamp	Connect SP to power source	Connect Service Point
	Install lamp	Install Item
	Attempt to notify owner - optional	Contact Customer
Connect SP	Connect SP to power source	Connect Service Point
Disconnect SP	Disconnect SP from power source	Disconnect Service Point
Install lamp	Install lamp	Install Item
Remove lamp and leave connected	Remove lamp	Remove Item
Install eye	Install eye	Turn On Item
Remove eye	Remove eye	Turn Off Item
Remove and disconnect lamp	Verify premise is vacant - optional	Contact Customer
	Remove lamp	Remove Item
	Disconnect SP from power source	Disconnect Service Point
Disconnect warning	Place disconnect warning	Contact Customer
Reconnect lamp	Install eye - optional	Turn On Item
	Install lamp - optional	Install Item
Cut lamp for non payment	Remove eye - optional	Turn Off Item

Examples of Unbadged Service Point Activity Types and their Steps

The following table shows several classic lamp-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Connect lamps in lot	Connect lamps	Reconfigure Multi-Item
Disconnect lamps in lot	Disconnect lamps	Reconfigure Multi-Item
Reconnect after pay	Reconnect lamps	Reconfigure Multi-Item
Disconnect warning	Place disconnect warning	Contact Customer
Cut lamps for non payment	Disconnect lamps	Reconfigure Multi-Item

Designing Other Field Activity Types

Besides those activity types that are needed to start / stop / cut service, you'll also need some field activity types for ad hoc service investigations and trouble orders.

The topics in this section illustrate these additional field activity types.

Examples of Meter-Oriented Service Investigation Activity Types and their Steps

The following table shows several classic meter-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Meter exchange	Remove meter	Remove Meter
	Define meter's retirement date	Change Meter
	Install meter	Install Meter
Investigate meter accuracy	Read meter	Read Meter
	Verify constant - optional	Change Meter's Configuration
	Read meter - optional	Read Meter
Meter service investigation order	Read meter - optional	Read Meter
	Check service point - optional	Change Service Point
	Remove meter - optional	Remove Meter
	Check meter attributes - optional	Change Meter
	Check constant - optional	Change Meter's Configuration
	Install meter - optional	Install Meter
	Contact customer - optional	Contact Customer

Examples of Lamp-Oriented Service Investigation Activity Types and their Steps

The following table shows several classic lamp-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Retire lamp and install another	Remove lamp	Remove Item
	Define lamp's retirement date	Change Item
	Install lamp	Install Item
Lamp service investigation order	Check service point - optional	Change Service Point
	Remove lamp - optional	Remove Item
	Check lamp attributes - optional	Change Item
	Install lamp - optional	Install Item
	Contact customer - optional	Contact Customer

Examples of Trouble Order Activity Types and their Steps

The following table shows trouble order-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Contact customer	Contact customer	Contact Customer

Examples of Device Testing Activity Types and their Steps

The following table shows device test-oriented activity types and their respective steps:

Activity Type	Step	Step Type Action
Bench test meter	Remove existing meter	Remove Meter
	Install new meter	Install Meter
	Test meter	Test Device
Field test meter	Remove existing meter	Remove Meter
	Test meter	Test Device
	Install new or existing meter	Install Meter
Bench test current transformer	Remove existing CT	Remove Item
	Install new CT	Install Item
	Test CT	Test Device

Field Activity Completion Considerations

The [Field Activity Step Upload Staging](#) table provided by the system supports the completion of the following "standard" step types:

- Connect Service Point
- Disconnect Service Point
- Install Meter
- Turn On Meter
- Turn Off Meter
- Read Meter
- Remove Meter
- Install Item
- Turn On Item
- Turn Off Item
- Remove Item

These step types are supported because the amount of information required to complete the step type is limited and is always the same for every step of that type. For example, to install a meter, the system must know the badge number and the effective date/time and a reading must exist for that date/time. Every Install Meter step requires the same information.

However, there are other step types whose completion information depends on what specifically occurred in the field. For example, if the meter configuration changed, there are several possible fields whose value may have changed such as dial format, read out type, full scale, etc. Because of the large number of possible fields that would need to be available to support completion of these "generic" step types, the field activity step upload staging table does not support capturing this information. The step types included in this category are the following:

- Change Item
- Change Meter
- Change Meter's Configuration
- Change Service Point
- Contact Customer
- Reconfigure Multi-Item
- Test Device

To complete field activities with one of the above step types, the recommendation is to use XAI to upload field activity completion information rather than using the [field order completion upload background processes](#). Using XAI, you can design a service to update the appropriate data in the system for the above step types and complete the field activity and all its steps.

NOTE:

Remove Meter. The system only allows meter reads marked as **Use on Bill** to be specified as removal reads. If your implementation wants to relax this validation and use reads that are not useable on bill as removal reads during FA completion, set the Removal MR Always Useable on Bill **Option Type** on the Meter Management Options [Feature Configuration](#) to Y.

FASTPATH:

Refer to [Field Activity Completion](#) for a sample service provided with the system to support completion for all step types.

Designing Field Service Classifications

When you set up a service point, you must define the operation area(s) that manage its fieldwork. If you have service points whose operation area differs based on the type of field activity, you will need multiple field service classifications (otherwise you'll just need 1 - call it All). For example, a service point may have one operation area for turn on / off field activities, a different operation area for trouble orders, and yet a third operation area for meter exchanges. We refer to each major category of service for which operation area differs as a Field Service Classification. In this example, you would need to define 3 field service classifications - On/Offs, Trouble Order, and Meter Exchanges.

NOTE:

Bottom line: if all types of activities at a given service point are dispatched from the same field office (i.e., operation area), you will only need one field service classification. If the field office of dispatch differs based on the type of field activity, you will need a field service classification for each category of field activity.

After you define your field service classifications, you need to associate them with your field activities (each field activity references a field service classification). You also need to define the field offices (i.e., operation areas) that perform work for each classification. For example,

- If trouble orders are dispatched from a central location, the Trouble Order service classification would have a single operation area linked to it.
- If turn ons / offs are dispatched from 4 separate operations area, you have the four operation areas linked to the On / Off field service classification.
- Etc.

If you're struggling with this concept, consider why the system needs to know about field service classifications:

- When a field activity is created, the system must associate it with an operation area. Why? Because operation area is one of the elements that controls the dispatch group to be associated with a field activity.
- The system finds the operation area by: a) extracting the field service classification from the field activity's activity type, and b) extracting the operation area for this classification defined on the service point.
- And finally, once the operation area is known, the system can allocate the dispatch group to the field activity. The other components that dictate the dispatch group are field activity type and service point type.
- Each service classification, in turn, would have its operations area defined.

NOTE:

How a service point gets its field service classifications and operation areas. A service point's field service classifications / operation areas will default based on its service type and its premise's postal code. See [Setting Up Premise & Service Point Postal Defaults](#) for more information.

Designing Who Does Your Field Activities

At this point, you have designed the following:

- Field activity types. These define what you do to your service points.
- SP types. These define the various services that exist at your premises.

- Operations areas. These define the locations from which you dispatch field activities.
- Dispatch groups. These define crews who do your field activities.

Now you have to pull it all together and define which dispatch group performs each field activity at every SP type in every area office. This is a four dimensional matrix that is easier to represent in two dimensions. We'll fill in this matrix for one of our many activity types. You'll need to do this for EVERY activity type:

Activity Type	SP Type	Operations Area	Dispatch Group
Connect SP and install meter	GAS - RESIDENTIAL	North Area	North - Gas&Water Crew
		South Area	South - Gas&Water Crew
	GAS - COMMERCIAL	North Area	North - Gas&Water Crew
		South Area	South - Gas&Water Crew
	WATER - RESIDENTIAL	North Area	North - Gas&Water Crew
		South Area	South - Gas&Water Crew
	WATER - COMMERCIAL	North Area	North - Gas&Water Crew
		South Area	South - Gas&Water Crew
	ELECTRIC - RESIDENTIAL	North Area	North - Electric Res Crew
		South Area	South - Electric Res Crew
	ELECTRIC - COMMERCIAL	North Area	North - Electric HV Crew
		South Area	South - Electric HV Crew

We made the following assumptions when filling in the above table:

- You have two operations areas: North Area and South Area.
 - The North Area handles all service requests in the northern part of your service territory.
 - The South Area handles all service requests in the southern part of your service territory.
- You have the following dispatch groups in the South Area.
 - North - Gas&Water Crew. This crew performs all field activities associated with ALL gas and water service points in the northern area regardless of SP type (i.e., residential and commercial installs are handled by the same group).
 - South - Gas&Water Crew. This crew performs all field activities associated with ALL gas and water service points in the southern area regardless of SP type (i.e., residential and commercial installs are handled by the same group).
 - North - Electric Res Crew. This crew performs all field activities associated with electric residential service points in the northern area.
 - South - Electric Res Crew. This crew performs all field activities associated with electric residential service points in the southern area.
 - North - Electric HV Crew. This crew performs all field activities associated with electric commercial service points in the northern area (the HV stands for high voltage).
 - South - Electric HV Crew. This crew performs all field activities associated with electric commercial service points in the southern area (the HV stands for high voltage).

You must fill in the above matrix for EVERY activity type. We'll give you one more example to show how to set up this information if you have centralized dispatching of some types of field activities. Our example assumes that all cuts for non-payment are handled by a single crew located at the North office.

Activity Type	SP Type	Operations Area	Dispatch Group
Cut metered service due to nonpayment	GAS - RESIDENTIAL	North Area	All - C&C crew

	South Area	All - C&C crew
GAS - COMMERCIAL	North Area	All - C&C crew
	South Area	All - C&C crew
WATER - RESIDENTIAL	North Area	All - C&C crew
	South Area	All - C&C crew
WATER - COMMERCIAL	North Area	All - C&C crew
	South Area	All - C&C crew
ELECTRIC - RESIDENTIAL	North Area	All - C&C crew
	South Area	All - C&C crew
ELECTRIC - COMMERCIAL	North Area	All - C&C crew
	South Area	All - C&C crew

We understand if you have the above situation, there is some redundancy. But this is the price of the flexible design.

At this point, you're ready to set up your field activity types and field activity type profiles. We recommend first setting up your field activity types. Then set up your field activity type profiles designed earlier.

NOTE:

Don't forget. A field activity type profile contains a list of ALL field activity types that can be performed at service points that use the profile. This list must include all of the field activity types needed to: start and stop service, cut and reconnect service, investigate service, and record trouble orders for the service. Refer to [Designing Your Field Activity Profiles & Types](#) for more information.

Setting Up Field Service Classification

To define your organization's service classifications and the locations at which each classification of service can be performed, open **Admin > Field Work > Field Service Class > Add**.

FASTPATH:

For more information about field service classifications, refer to [Designing Field Service Classifications](#).

Description of Page

Enter an easily recognizable **Field Service Class** and **Description** for each service classification.

The **Operation Area** collection shows the operation areas in which field activities associated with a service classification are dispatched.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_FS_CL](#).

Setting Up Field Activity Types

The topics in this section describe how to set up field activity types.

NOTE:

When a new activity type is added. When you introduce a new field activity type, you must define the field activity type profiles on which the activity can be performed. Refer to [Setting Up Field Activity Type Profiles](#) for more information. If the field activity type is "dispatchable", you must set up the rules used by the system to default the dispatch group onto field activities of this type. You do this on [Field Service Control](#).

FASTPATH:

For more information about field activity types, refer to [Designing Your Field Activity Profiles & Types](#).

Field Activity Type - Main

You begin to define a field activity type by opening **Admin > Field Work > Field Activity Type > Add**.

Description of Page

Enter an easily recognizable **Activity Type** and **Description** for each field activity type.

Use **Field Activity Priority** to define the priority associated with field activities of a given type. This priority affects the order in which field activities appear on the dispatching inquiries. Important activities (e.g., trouble orders) should have a higher priority than less important activities.

NOTE:

The values for this field are customizable using the Lookup table. This field name is FA_PRIORITY_FLG.

Use **Field Service Class** to define the field activity's category of service. Refer to [Designing Field Service Classifications](#) for information about the significance of this field.

If this FA type is for activities that will be synced to service orders from an external system, specify the external system in **Fieldwork Orchestration**.

Turn on **Eligible for Dispatch** if this type of field activity is dispatched to a field crew. This switch will be on for all field activities except the one used to indicate the system should use the next scheduled meter read as the initial or final read on a service agreement. If this switch is on you must set up the rules used by the system to default the dispatch group onto field activities of this type. You do this on [Field Service Classification](#).

If your FA type is eligible for dispatch and the [Appointments](#) module is not **turned off**, indicate if **Appointment Booking** for field activities of this type is Required for Dispatch, Optional or Not Applicable. If eligible for dispatch is checked, this value is defaulted to optional ; otherwise it is defaulted to not applicable .

Turn on **Display as Alert** if Control Central should display an alert if its premise has a completed field activity of this type. If this switch is on,

- Use **Nbr Days Alert Active** to define the number of days the alert should appear on Control Central. The field activity's scheduled date is used as the start date for the alert period.
 - Enter the **Alert Information** to appear on Control Central.
-

NOTE:

Recommendation. We recommend only using this feature on unusual field activity types (e.g., disconnect warnings, cut for non-payments) so that a CSR is not presented with an alert for every field activity type.

Enter the **Business Object** for this FA Type if the FA type will need access to the BO plug-in spots. If the FA type has a business object specified, the system applies the business object's rules when the FA is added, changed, or deleted. This includes the business object's validation algorithms (only executed after the "core validation" specified on the maintenance object is done), post-processing algorithms, and audit algorithms. The maintenance object for the FA should specify an

algorithm that finds its business object using its FA Type. Refer to the base package Determine Standard Business Object algorithm for an example of this type of algorithm.

Use the **Field Activity Step** collection to define the discreet actions involved in the execution of the activity. Keep in mind that an activity type's steps are used to:

- Provide guidance to the field staff in respect of the expected steps involved in the execution of the activity.
- Simplify navigation to the page groups used to record what actually took place in the field.

The **Step Sequence** is system-assigned and may not be modified.

Enter the step's **Description**. This information is printed on the field order.

Indicate whether the step is **Optional**. This indicator is used when the user attempts to complete the field activity (after the fieldwork is complete). If an activity contains required steps, the system will not allow the activity to be completed unless every required step has an indication of what happened. For example, if an activity contains a step indicating the meter must be read, the activity cannot be completed until the meter read is referenced on the activity.

Use **FA Step Type Action** to define the activity associated with the step. The action determines the foreign key that must be referenced on the step in order to complete it. For example, the Read Meter action requires a meter read id to be linked to the step in order to complete the activity. The permissible values are defined in the following table.

Step Type Action	What it's used for
Change Item	Used when the activity changes attributes on an item (e.g., the date retired).
Change Meter	Used when the activity changes attributes on a meter (e.g., the date retired)
Change Meter's Configuration	Used when the activity changes attributes on a meter's registers (e.g., the constant)
Change Service Point	Used when the activity changes attributes on service point (e.g., the location)
Connect Service Point	Used when the activity connects a service point to its source (e.g., connecting an electric service point to the source of electricity)
Contact Customer	Used when the activity involves contact with a customer
Disconnect Service Point	Used when the activity disconnects a service point from its source (e.g., disconnecting an electric service point from the source of electricity)
Install Item	Used when the activity installs a badged item at a service point
Install Meter	Used when the activity installs a meter at a service point
Read Meter	Used when the activity reads a meter
Reconfigure Multi-Item	Used when the activity changes the number / type of unbadged items at a service point
Remove Item	Used when the activity removes a badged item from a service point
Remove Meter	Used when the activity removes a meter from a service point
Test Device	Used when the activity tests a meter or item.
Turn Off Item	Used when the activity turns off a badged item at a service point
Turn Off Meter	Used when the activity turns off a meter at a service point
Turn On Item	Used when the activity turns on a badged item at a service point
Turn On Meter	Used when the activity turns on a meter at a service point

Use the **Characteristic** collection to define **Characteristic Types** and their respective **Characteristic Values** to describe characteristics that are common to all field activities of this type.

NOTE:

You can only choose characteristic types defined as permissible on the field activity type record.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_FA_TYPE](#).

Field Activity Type - FA Characteristics

To define characteristics that can be defined for field activities of a given type, open **Admin > Field Activity Type > Search** and then navigate to the **FA Characteristics** page.

Description of Page

Use the **Characteristics** collection to define characteristics that can be defined for field activities of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on field activities of a given type. Turn on the **Default** switch to default the **Characteristic Type** when field activities of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

NOTE:

Field activities created by the system. When setting the Required switch, remember that most field activities are created by the system. Only turn on the required switch for these types of activities if a default characteristic value can also be indicated

Field Activity Type - FA Completion Control

Open **Admin > Field Work > Field Activity Type > Search** and then navigate to the **FA Completion Control** page to define special functions that should be executed when a field activity is completed.

Description of Page

NOTE:

These algorithms are optional. The use of completion algorithms on a field activity type is optional. You would only use them if you have special functions that should be executed when a given field activity type is completed. Read the information below for examples.

The **Field Activity Completion Controls** tab is used when an algorithm should be executed when a field activity is completed. For example, if a charge should be levied when a certain type of activity is completed, you would indicate the "levy adjustment" algorithm should be executed. The type of algorithm may differ based on the CIS division in which the service point's premise is located. The following fields must be defined:

CIS Division Defines the division associated for which the algorithm will be executed. The system will only execute the algorithm when a field activity is performed at a service point whose premise is governed by the division.

FA Completion Algorithm Defines the algorithm that will be executed when a field activity is performed at a service point whose premise is governed by the associated division. Click [here](#) to see the algorithm types available for this plug-in spot.

Field Activity Type - SP Type Review

Open **Admin > Field Work > Field Activity Type > Search** and then navigate to the **SP Type Review** page to review the SP types at which the field activity can be performed. And for each SP type, you can view the dispatch group that will perform the activity at every operations area.

NOTE:

Four dimensions. For every **field activity type**, you define the **dispatch group** that performs the activity at every **SP type** located in every **operations area**. This information is maintained on the Field Activity Type page group. This is a rather complex relationship because it involves the four dimensions highlighted in bold. Due to this complexity, we have provided review trees on the SP Type, Dispatch Group, and Field Activity Type pages to help you understand what you've set up.

Description of Page

This page is dedicated to a tree that shows the SP types at which the field activity can be performed. And for each SP type, you can view the dispatch group that will perform the activity at your operations areas.

Setting Up Field Service Control

Open **Admin > Field Work > Field Service Control > Add** to define the dispatch group that is responsible to perform a field activity in each operation area for each SP type.

Description of Page

NOTE:

Eligible for dispatch. You would only define this information for field activity types that are eligible for dispatch because these are the only ones that need dispatch groups.

If a field activity type is **Eligible for Dispatch**, you must define the **Default Dispatch Group** that will be assigned to field activities. You do this in respect of the **Field Activity Type**, the field activity's service point's **SP Type**, and the service point's **Operation Area**. In addition you must define the following for every combination of **Activity Type**, **SP Type** and **Operation Area** you define the following:

Priority The priority controls the order in which the system calls the algorithms that determine the **Dispatch Group** to be assigned to field activities associated with a given **Activity Type**, **SP Type** and **Operation Area**. Higher priorities are used before lower priorities.

Dispatch Algorithm Select the algorithm that determines the dispatch group to be assigned to the field activity.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that assigns a dispatch group. Click [here](#) to see the algorithm types available for this plug-in spot.

IMPORTANT:

You must have at least one entry in this collection otherwise the system will not assign a dispatch group to a field activity. This entry should have the lowest priority code and should reference a **Dispatch Algorithm** that references the **OFSDGRP DFLT** algorithm type.

Where Used

This information is used to default the appropriate dispatch group on new field activities.

Setting Up Field Activity Type Profiles

The topics in this section describe the pages used to define a field activity type profile.

NOTE:

When a new profile is added. When you introduce a new field activity type profile, you must define the SP types that use it. Refer to [SP Type - Main](#) for more information.

FASTPATH:

For more information about field activity type profiles, refer to [Designing Your Field Activity Profiles & Types](#).

Field Activity Type Profile - Main

Open **Admin > Field Work > Field Activity Type Profile > Search** to maintain a field activity type profile.

Description of Page

Enter a unique **Field Activity Type Profile** and **Description** for the activity type profile.

The **FA Type Profile Templates** indicate the templates that exist for this field activity type profile. Use the drill down button to go directly to the desired template. Alternatively, you can go to the **Template** tab and scroll until you find the correct template.

Field Activity Type Profile - Template

Open **Admin > Field Work > Field Activity Type Profile > Search** and then navigate to the **Template** page to define the field activity(s) used for various situations in the system.

Description of Page

The information in the **Field Activity Profile Template** collection defines the field activity(s) created for each situation identified by the **Customer Event**. The possible customer events are Cut for Non-payment, Disconnect Warning, Reconnect for Payment, Reread, Stop Service, Start Service, and Start/Stop. Other customer events can be defined on the [Look Up](#) page (search for the CUST_EVT_FLG field name). Refer to the following sections for more detail about customer events included with the base product.

The fields defined for each event are common. You define the field activity(s) to be generated given the condition of the service point and the location at which service was disconnected (when applicable). The following fields display:

SP Field Condition Define the condition of the service point associated with the field activity. Valid values correspond to those described under [Typical Metered Service Matrix](#), [Typical Badged Item Service Matrix](#) and [Typical Unbadged Item Service Matrix](#).

Sequence You will typically have a single field activity for any specific combination of SP Field Condition and Disconnect Location. Therefore you'll just have a single sequence (say 10) for each combination. If you need to generate multiple field activities based on a given combination, use a unique sequence number for each activity.

No Activity Turn on this switch if no field activity should be generated for the condition. This is typically off for non-metered (e.g., lamp) starts when the service is already started.

Disconnect Location The type of field activity can differ based on where the service point was disconnected from the source of service. For example, if it the SP is disconnected at the pole, you would have a different field activity than if it is disconnected at the meter.

Activity Type Define the type of activity to be generated. You should take care to use activity types defined as valid for the SP type (on the last page).

Where Used

Refer to the following sections for information about where each template is used.

Field Activities Initiated To Start Service

Use the customer event Start to define the field activity(s) used to start service at a service point whose SP type references this profile.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

The start service process uses this information to determine the type of field activities to create to start service at a service point.

WARNING:

Warning. Field activities will only be created for starts if you have defined the appropriate field work creation algorithm on the service agreement's SA type. Refer to [SA Type - Algorithms](#) for more information.

Field Activities Initiated To Stop Service

Use the customer event Stop to define the field activity(s) used to stop service at a service point whose SP type uses this profile.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

The stop service process uses this information to determine the type of field activities to create to stop service at a service point.

WARNING:

Warning. Field activities will only be created for stops if you have defined the appropriate field work creation algorithm on the service agreement's SA type. Refer to [SA Type - Algorithms](#) for more information.

Field Activities Initiated For Back-to-Back Service

Use the customer event Start/Stop to define the field activity(s) used to stop service for one customer and start service for another at a service point whose SP type uses this profile.

NOTE:

Terminology. We use the term **back-to-back** to describe the situation when a single field activity supports both the stop and start service requests. The system sets up a back-to-back situation by default when it is aware of both the start and stop customers at a premise.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

The start/stop service process uses this information to determine the type of field activities to create to start service at a service point.

Field Activities Initiated To Cut Service Due To Non-Payment

Use the customer event Cut for Non-Payment to define the field activity(s) used to cut service at a service point whose SP type uses this profile.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

Severance and cut events use this information to determine the type of field activities to create to cut service at a service point.

Field Activities Initiated To Place A Disconnect Warning At A Service Point

Use the customer event Disconnect Warning to define the field activity(s) used to place a disconnect warning at a service point whose SP type uses this profile.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

Severance and cut events use this information to determine the type of field activities to create to leave a disconnect warning at a service point.

Field Activities Initiated To Reconnect Service At A Service Point

Use the customer event Reconnect for Payment to define the field activity(s) used to reconnect service (after being cut) at a service point whose SP type uses this profile.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

Severance and cut events use this information to determine the type of field activities to create to reconnect service at a service point.

Field Activities Initiated To Reread A Meter At A Service Point

Use the customer event Reread to define the field activity(s) used to reread a meter located at a service point whose SP type uses this profile.

The field activity type, M-REREAD, is used by the [meter read](#) page to create a field activity when a user requests a meter to be reread.

Description of Page

See [Field Activity Type Profile - Template](#) for a description of the fields.

Where Used

The [meter read](#) page uses this information to determine the type of field activity to create to reread a meter.

Defining A Profile's Valid Field Activity Types

Open **Admin > Field Work > Field Activity Type Profile > Search** and choose the **Type** page to define the superset of field activity types that may be performed on service points whose SP type uses this profile.

Description of Page

The **Field Activity Type** collection shows the field activities that may be performed at service points whose SP type references the field activity type profile.

Where Used

This information is used to control the types of field activities that may be performed at a service point.

Setting Up Fieldwork Cancellation Reasons

When you cancel a field activity, you must supply a cancellation reason. To define fieldwork cancellation reasons, open **Admin > Field Work > Fieldwork Cancel Reason**.

Description of Page

Enter a **Cancel Reason** and **Description** for every field activity/field order cancellation reason.

Only use **System Default** on those reason codes that are placed on field activities that are automatically canceled by the system. The following table lists the valid values and the condition where this cancel reason is used.

System Default	System Condition
Cut Process canceled	Placed on field activities that are canceled when a Cut Process is canceled.
Device Test Selection canceled	Placed on field activities that are canceled when a Device Test Selection is canceled.
Near MR FA Completion canceled	Placed on field activities that use a scheduled meter read to start / stop service when they are canceled. Refer to How To Start Service Using A Scheduled Meter Read for more information about these special field activities and how they may be canceled.
SA Start/Stop canceled	Placed on field activities that are canceled when a pending start / stop is canceled.

NOTE:

Required values. You must have one reason code defined for each of the System Default values that corresponds to an event that may occur in your implementation.

Feature Configuration. Some organizations require a cancel reason to be specified when a field order is cancelled. To achieve this, you must set up a fieldwork options [feature configuration](#) and ensure that the Cancel Reason Required option is set to Y.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_FA_CAN_RSN](#).

Fieldwork Reschedule Reason

Some organizations require a reschedule reason to be specified when a field activity or field order's schedule date/time is changed. To achieve this, you must set up a fieldwork options [feature configuration](#) and ensure that the Reschedule Reason Required option is set to Y. The Default Reschedule Reason option should also be specified. This option value will be used when the system updates a field activity's schedule date/time behind the scenes. To define reschedule reasons, open **Admin > Field Work > Fieldwork Reschedule Reason**.

Description of Page

Enter a **Reschedule Reason** and **Description** for every field activity/field order reschedule reason.

FASTPATH:

For more information on how to audit changes to a field activity or field order's schedule date/time, refer to the Field Activity Rescheduling - Audit and Field Order Rescheduling - Audit business objects.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_FA_RESCHED_RSN](#).

Setting Up Field Activity Remarks

You may link remarks to a field activity using remark codes. To define field activity remark codes, open **Admin > Field Work > Field Activity Remark > Add**.

Description of Page

Enter a unique **Field Activity Remark** and a **Description** for every field activity remark.

Turn on **Eligible for Processing** if field activities marked with a given remark code should cause one or more algorithm to execute.

The grid contains **Algorithms** associated with the field activity remark. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Field Activity Remark Activation	Optional	<p>These algorithms are executed when there are pending FA remarks linked to a field activity and the FACT (Field activity remark activation) background process runs.</p> <p>Refer to Field Activity - Characteristics/Remarks for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_FA_REM_CD](#).

Setting Up Outage Call Types

When you create an outage call, you must supply an outage call type. Outage call types contain a great deal of information that is defaulted onto the outage call, including the outage category group codes or trouble codes. To set up outage call types, open **Admin > Field Work > Outage Call Type > Add**.

FASTPATH:

Refer to [The Big Picture of Outage System Integration](#) for a detailed description of how trouble calls are created and sent to NMS.

The topics in this section describe the base-package zones that appear on the Outage Call Type portal.

Outage Call Type List

The Outage Call Type [List zone](#) lists every outage call type, i.e. every service task type that has a service task type class of Outage Call. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent outage call type.
- Click the Add link in the zone's title bar to add a new outage call type.

Outage Call Type

The Outage Call Type zone contains display-only information about an outage call type. This zone appears when an outage call type has been broadcast from the Outage Call Type [List zone](#) or if this portal is opened via a drill down from another page. The following functions are available:

- **Edit**: to start a business process that updates the outage call type.
- **Delete**: start a business process that deletes the outage call type.
- **Duplicate**: to start a business process that duplicates the outage call type.

Please see the zone's help text for information about this zone's fields.

Defining Credit & Collections Options

NOTE:

The functionality described in this section is meant to handle the collection of unpaid balances. If your organization practices [open-item accounting](#) and collects on unpaid bills, you will not use this functionality. Rather, you will use the functionality described under [Defining Overdue Processing Options](#).

The system periodically monitors how much your customers owe to ensure they haven't violated your collection criteria. When a violation is detected, the system generates the appropriate responses (e.g., letters, disconnect notices, collection agency referrals, and eventually write off). This section describes how to set up the tables that control your credit & collections processing.

WARNING:

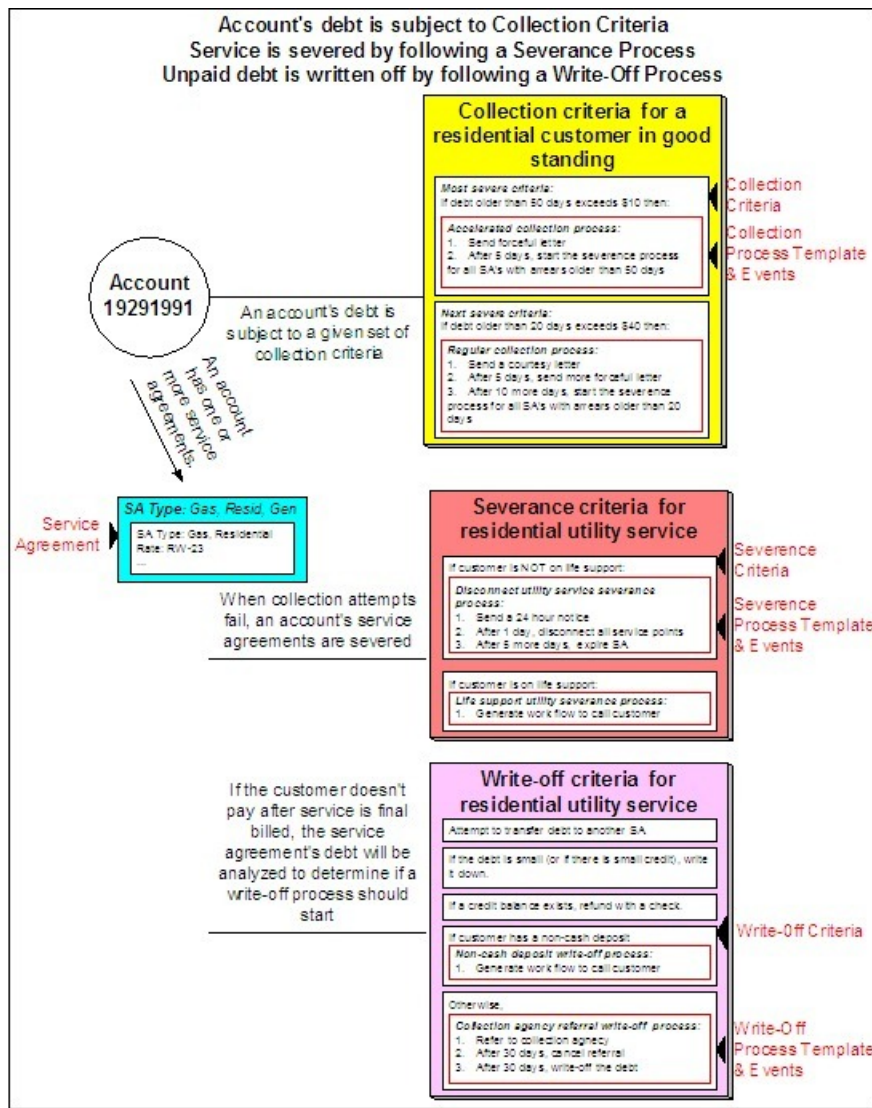
Setting up the credit & collections control tables is as challenging as your organization's collection rules. If you have simple rules then your setup process will be straightforward. If your collection rules are complicated (e.g., they differ based on the type of customer, the type of debt, the age of debt, the type of service, etc.), then your setup process will be more challenging.

The Big Picture Of Credit & Collections

This section provides an overview of important C&C concepts with which you should be familiar before you set up your C&C control tables.

Collection Criteria vs. Severance Criteria vs. Write Off Criteria

The following diagram introduces important concepts related to the C&C processes:



There are many important concepts illustrated above:

An account's debt comes from its service agreements

An account's debt is managed at the service agreement level, i.e., the system keeps track of how much a customer owes in respect of each service agreement. In order to determine an account's balance, the system must add up the debt on each of the account's service agreements.

Collection criteria define intolerable debt

Collection criteria are control data that define intolerable debt. Most criteria are defined using a combination of number of days in arrears and a dollar amount.

Collection criteria may be compared to an account's total debt or to subsets of debt

If your organization has simple collection procedures, you will probably target collection criteria at an account's total debt. However, you have the option of segregating an account's debt into debt classes and targeting the collection criteria at each class. For more information about debt classes, see [Different Collection Criteria For Different Customers And Different Debt](#).

Collection criteria also define what to do when the level of intolerable debt is exceeded

When you define collection criteria, you also define how the system should respond if an account violates your criteria. These collection events are defined in respect of a "collection process template".

There are usually several collection events that take place when criteria are violated

A collection process template usually has several collection events. Each event is meant to prod the customer to pay. The initial collection events are typically letters. If payment is not received after several such attempts, the last collection event typically starts a severance process for each service agreement in arrears.

A severance process template defines how to sever a service agreement

A "severance process template" defines how to sever a given type of service agreement. A severance process template usually contains several severance events. These events are a series of letters and / or disconnection field activities that eventually result in the expiration of a service agreement if payment is not received.

Severance criteria define how to sever service agreements

Severance criteria define the severance process to be executed for service agreements of a given SA type. The severance process may differ depending on some attribute of the customer (or premise). For example, you may have a different severance process if the customer has life support equipment.

After a service agreement is severed, it will be final billed

When the last active service agreement linked to an account is stopped, the system changes the account's bill cycle to bill that evening. If only one of many SAs is stopped, the SA will only be final billed as per the account's original bill cycle schedule.

If a customer doesn't pay their final bill, the account's debt will be analyzed to determine if the system can reduce the debt to zero using a variety of mechanisms

The system will look at an account's finaled debt on its next scheduled credit review date (typically a few days after the bill's due date). The system will attempt to reduce the service agreement's debt to zero using all of the following methods:

- If the account has active service agreements, it will transfer the finaled debt to an active service agreement.
- If the debt or credit amount on the service agreement is small, the system will generate an adjustment to 'write it down' (or up in the case of a small credit).
- If the service agreement has a large credit amount, the system will generate an A/P adjustment (resulting in a check being sent to the customer).

If a customer's finaled debt cannot be reduced via any of the previous methods, the system creates a write-off process

A write-off process contains one or more write-off events. These events can generate a letter, send a To Do entry to a CSR, send a referral to a collection agency, etc.

When you set up the system, you define the type of write-off process to use for every collection class / write-off debt class combination. In addition, you can also indicate when the type of write-off process should differ depending on some attribute of the customer (or premise). For example, you may have a different write-off process if the customer has a non-cash deposit.

The last write-off event typically causes the debt to be written off

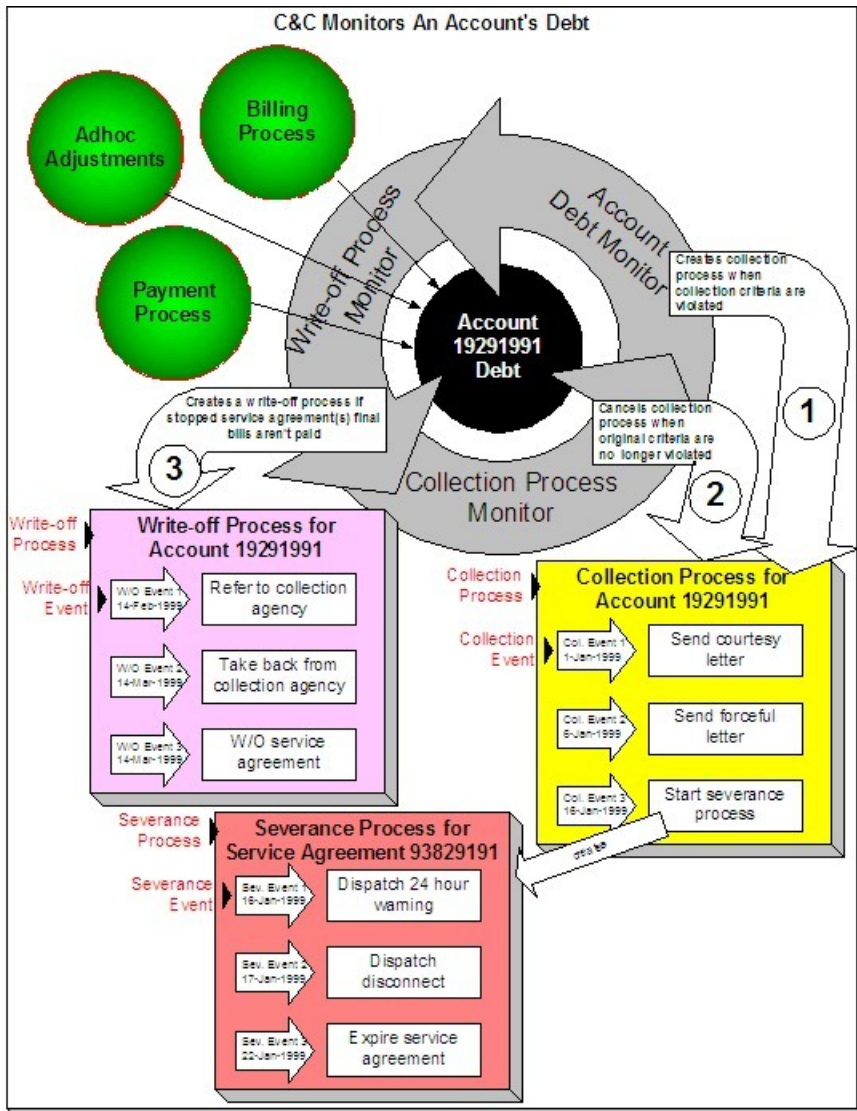
Ultimately, if the write-off events fail, the debt will have to be written off. When debt is written-off, the system creates a write-off service agreement and transfers the outstanding debt to it. This means the debt stays with the account for life and will have to be paid off if the customer ever returns.

NOTE:

Checkpoint. At this point, you should be familiar with the concept that an account's debt is compared to user-defined collection criteria. If the account violates the criteria, a series of events will ensue that prod the customer to pay. If the customer doesn't respond, every service agreement in arrears will be severed (i.e., disconnected). If lack of service doesn't inspire payment, the service agreement will be expired and a write-off process will be created to manage the write-off activities.

The C&C Monitors

Your collection, severance and write-off criteria described in the previous section exist to support the processes that manage the collection of overdue debt. The following diagram illustrates, at a high level, the major processes that manage the collection of overdue debt:



There are many important concepts illustrated above:

Bills, payments and adjustments affect an account's debt	An account's debt is the accumulation of all bills, payments and adjustments.
The Account Debt Monitor creates a collection process when an account violates collection criteria	Periodically, a background process referred to as the Account Debt Monitor (ADM and ADM2) determines if an account's debt violates your collection criteria. If so, a collection process is created using the violated criteria's collection process template. Refer to When Is An Account's Debt Monitored? for a description of when an account's debt is compared against collection criteria.
A collection process contains one or more collection events	The collection process contains a series of collection events. These events correspond with the collection event types associated with the collection process template. The initial collection events are typically letters. If payment is not received after several such communications, the last collection event typically starts a severance process for each service agreement in arrears.
The Collection Process Monitor cancels a collection process when warranted	The Collection Process Monitor cancels a collection process when its service agreements satisfy your cancellation criteria (e.g., when the service agreements have less than \$10 of debt older than 20 days). Refer to How Are Collection Processes Cancelled for more information about the cancellation process.
The last collection event starts one or more severance processes	The last collection event typically starts one or more severance processes. A severance process contains the activities necessary to sever a service agreement. The service agreement(s) that are severed may be all SAs that are associated with the collection process. Alternatively, you can nominate a service agreement to act as the primary service to cut (you'd do this if you cut electricity when the customer doesn't pay for their gas). The algorithm on the collection event that starts severance will control which service agreement(s) are severed. Refer to How To Nominate A Single Service Agreement To Sever for more information.
Each service agreement has its own severance process	Every service agreement that is severed has a severance process. The type of process is dependent on the severance criteria linked to the service agreement's SA type.
A severance process contains one or more severance events	The severance process contains a series of severance events. The events correspond with the severance process template's severance events.
The system cancels a severance process when warranted	The system cancels a severance process when its service agreement satisfies your cancellation criteria (note, it is possible to set up the system so that all service agreements in the debt class must satisfy your cancellation criteria before a severance process is cancelled). It's important to note that the cancellation is real time (as opposed to the cancellation of collection processes, which happens in a background process). Refer to How Are Severance Events Canceled? for more information.
The last severance event should expire the service agreement	The last severance event typically expires its service agreement. When the last service agreement linked to an account is expired, the system will schedule the account for billing (outside of its normal bill cycle schedule).
If you nominate a single SA to sever when multiple SAs are in arrears...	Earlier we indicated that you can nominate a service agreement to act as the primary service to cut (you'd do this if you cut electricity when the customer doesn't pay for their gas). If you do this, you also need

a severance event that will sever all other service agreements in the debt class if the severance of the nominated service agreement doesn't inspire payment. A severance event algorithm to do such is supplied with the base package. Refer to [How To Nominate A Single Service Agreement To Sever](#) for more information.

The Write-Off Monitor creates a write off process to collect stopped, unpaid debt

The Write-Off Monitor reviews stopped and reactivated service agreements after their closing bill's due date (plus grace period). The Write-Off Monitor attempts to reduce the service agreement's debt to zero using all of the following methods:

- If the account has active service agreements, it will transfer the finalized debt to an active service agreement.
- If the debt or credit amount on the service agreement is small, the system will generate an adjustment to 'write it down' (or up in the case of a small credit).
- If the service agreement has a large credit amount, the system will generate an A/P adjustment (resulting in a check being sent to the customer).
If the system is unsuccessful in reducing the account's debt to zero, a write-off process will be created using the appropriate write-off process template. Refer to [The Big Picture Of Write Off Processing](#) for more information about the write-off process.

A write-off process contains one or more write-off events

The write-off process contains a series of write-off events. These events correspond with the write-off event types associated with the write-off process template. The initial write-off events are typically collection agency referrals and/or letters. If payment is not received as a result of such efforts, the last write-off event typically writes off the customer's debt.

The system cancels a write-off process when warranted

The system cancels a write-off process when its service agreements no longer have debt (i.e., they become closed).

Another write-off process will be created if a closed service agreement ever reactivates

If a service agreement becomes reactivated (e.g., because the final payment bounces), the service agreement will be processed by the Write-Off Monitor and the whole write-off process starts again.

NOTE:

Checkpoint. At this point, you should be familiar with the concept that a collection process will be created for an account that violates collection criteria. The collection process consists of a series of events that typically generate letters and / or To Do entries. If the customer doesn't respond, a severance process will be started for one or more service agreements. A severance process consists of a series of events that typically generate letters and/or disconnection field activities. If lack of service doesn't inspire payment, the last severance event expires the service agreement (and a final bill will be scheduled when the last service agreement is expired). If the customer doesn't pay the final bill, a write-off process will be created for each type of unpaid debt. The write-off process consists of a series of events that ultimately result in the write-off of the customer's debt. When debt is written-off, the system creates a write-off service agreement and transfers the outstanding debt to it. This means the debt stays with the account for life (because the write-off service agreement is linked to the account) and will have to be paid off if the customer ever returns.

The Big Picture Of Collection Processes

The topics in this section describe how collection processes are created and cancelled.

FASTPATH:

For more information refer to [The Lifecycle Of A Collection Process And Its Events](#).

How Does The Account Debt Monitor Work?

This section describes how the Account Debt Monitor uses your collection criteria and collection process templates to collect overdue debt.

Different Collection Criteria For Different Customers And Different Debt

Consider the following:

- You probably have different collection criteria for different jurisdictions (i.e., CIS Divisions). For example, if you have customers in different states / provinces, you may have different regulator-imposed criteria applied to each state's debt. You differentiate your debt in respect of the collection process via the **CIS division code on each customer's account**.
- You probably have different collection criteria for different customer segments. For example, customers with large bills probably have strict criteria, whereas you're probably more lenient with small customers (or vice versa). You differentiate your customers in respect of the collection process via a **collection class code on the customers' accounts**. An account's initial collection class is defaulted from its customer class. You may override an account's collection class at will.
- You probably have different collection criteria for different classes of debt. For example, if a single customer has both regulated and unregulated debt, you probably have commission-imposed criteria to be applied to the regulated debt, but you have the freedom to apply stricter criteria to the unregulated debt. You differentiate your debt in respect of the collection process via a **debt class code on the customers' service agreements** (note: the debt class is actually defined on the service agreement's SA type).
- You will have different criteria for every currency in which you work because the monitoring process always compares a customer's debt against some value and this value must be denominated in the customer's currency. A customer's currency is defined using a **currency code on the account**.

Given the above, you should understand that different collection criteria will exist for every combination of CIS division, collection class, debt class, and currency code. If you're confused, consider the following matrix (where we assume you have a single currency and division and therefore avoid the third and fourth dimensions):

SA's Debt Class	Account's Collection Class:	
	Commercial Customer	Residential Customer
Regulated	N/A - there is no regulated, commercial customer debt.	Highest Priority: If > \$5 in arrears by more than 50 days, create the accelerated collection process for residential customers. Lower Priority: If > \$25 in arrears by more than 25 days, create the courtesy reminder collection process for residential customers.
Unregulated	Highest Priority: If > \$10 in arrears by more than 50 days, create the accelerated collection process for commercial customers. Lower Priority: If > \$1000 in arrears by more than 25 days, create the normal collection process for commercial customers.	Highest Priority: If > \$10 in arrears by more than 25 days, create the normal collection process for residential customers.

Charitable Contribution

Highest Priority: If > \$10 in arrears by more than 50 days, create the charitable collection process.

Highest Priority: If > \$10 in arrears by more than 50 days, create the charitable collection process.

Also, notice that there can be multiple criteria for each cell in the matrix. What differentiates one collection criteria from another is its priority. The higher priority criteria will be compared first. If the debt meets the criteria, the collection process is initiated and no further comparisons are performed.

FASTPATH:

For more information about maintaining this matrix, refer to [Setting Up Collection Class Controls](#). For more information about how the system handles an element in this matrix that has multiple criteria, see [How Is An Account's Debt Monitored?](#).

Override Conditions

WARNING:

Your credit & collection requirements may not require any overrides and therefore this section may not be relevant for your organization.

The matrix presented in the previous section showed:

- You can have different collection criteria for different categories of debt and customers.
- When a collection criteria is violated, the system generates a specific collection process.

This works great for many organizations, but if your organization has other factors that affect either the collection criteria OR the collection process that is initiated when the criteria is violated, you may need to use override collection criteria. For example,

- If you have a different collection process for regulated, residential debt during the winter months, you'll need to use override collection criteria (where the override criteria is "if it's winter").
 - If you have different collection criteria for customers with a poor credit score, you'll need to use override collection criteria (where the override criteria is "if the customer's credit rating is poor").
-

FASTPATH:

Refer to [Designing Your Collection Class Control Overrides](#) for more information.

This section describes how and when the Account Debt Monitor analyzes an account's debt.

When Is An Account's Debt Monitored?

The account debt monitor (ADM) analyzes an account's debt at least every X days, where X is defined on the [customer class control](#) associated with the account's customer class and division (in the field Min Credit Review Freq (Days)).

In addition, an account's debt will also be monitored as follows:

- The ADM looks at an account's debt X days after an account's bill due date (X is defined on the account's customer class in the field Collection Grace Days).
- The ADM looks at an account's debt after a payment is canceled when the cancellation reason indicates NSF (non-sufficient funds).

- The ADM looks at an account's debt after a payment arrangement is broken (assuming you use the base package's break payment arrangement plug-in). Refer to [Monitoring Payment Arrangements](#) for more information.
- The ADM looks at an account's debt after a pay plan is broken. Refer to [The Pay Plan Monitor](#) for more information.

How Is An Account's Debt Monitored?

Assume the following collection control matrix exists for your organization:

SA's Debt Class	Account's Collection Class:	Account's Collection Class:
	Commercial Customer	Residential Customer
Regulated	N/A - there is no regulated, large customer debt	Highest Priority: If > \$5 in arrears by more than 50 days, create the accelerated collection process for residential customers. Lower Priority: If > \$25 in arrears by more than 25 days, create the courtesy reminder collection process for residential customers.
Unregulated	Highest Priority: If > \$10 in arrears by more than 50 days, create the accelerated collection process for commercial customers. Lower Priority: If > \$1000 in arrears by more than 25 days, create the normal collection process for commercial customers.	Highest Priority: If > \$10 in arrears by more than 25 days, create the normal collection process for residential customers.

This matrix contains the information used by the Account Debt Monitor.

FASTPATH:

For more information about the information in this matrix, refer to [Different Collection Criteria For Different Customers And Different Debt](#).

This matrix can be overwhelming when viewed as a whole. So let's consider how to use it for a specific account's debt and things will become clearer.

First, because an account belongs to a unique collection class, we only have to worry about a single column in the matrix when monitoring an account's debt.

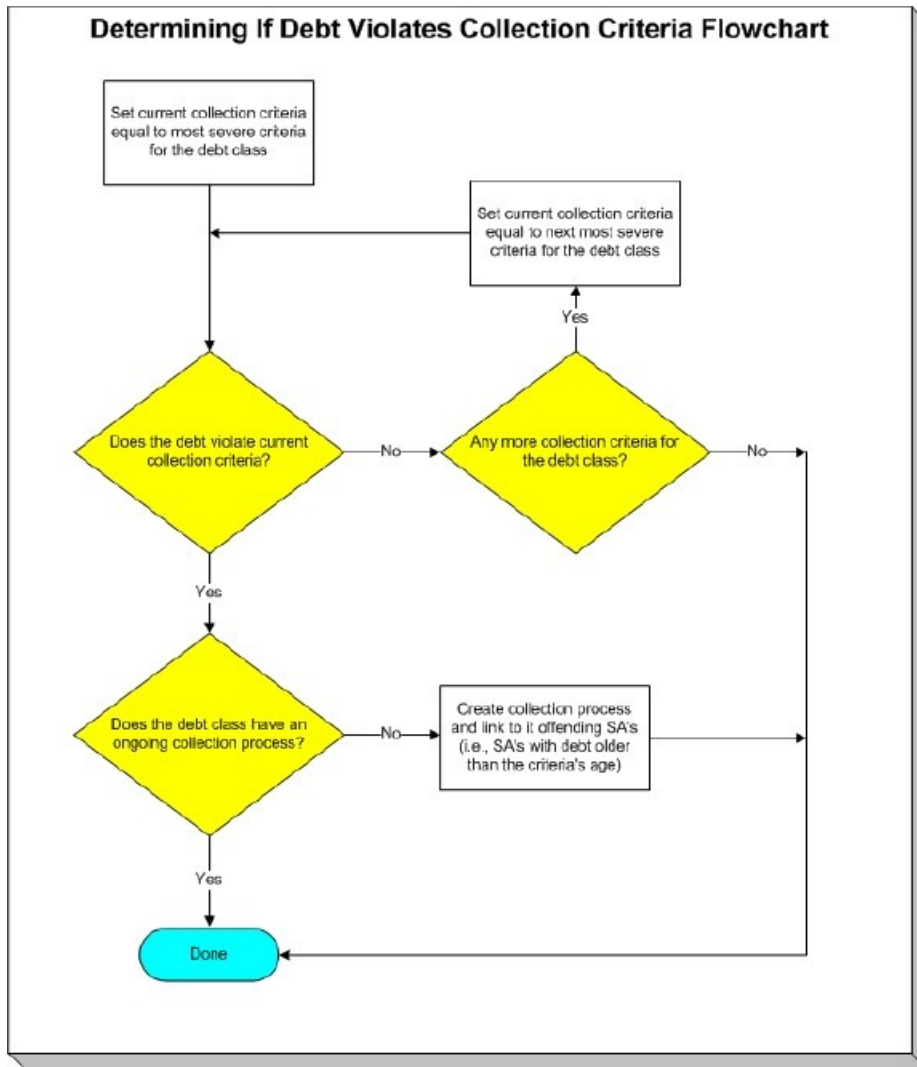
Next, we accumulate the total amount of aged debt for each unique debt class associated with the account's service agreements.

Next, we subject the accumulated aged debt to the override aged debt algorithm (plugged in on the debt class). This algorithm can cause aged debt to be reduced. This is an optional algorithm and is only used if you set up pay plans for customers. Refer to [How Pay Plans Affect The ADM](#) for more information.

Next, we determine if the debt for the debt class violates the collection criteria in the respective matrix element. If so, we kick off a collection process and link the offending service agreements to it. The logic associated with the determination of whether to kick off a collection process is rather sophisticated. The following flowchart explains the exact details.

NOTE:

Important. If a service agreement is part of an ongoing severance process, it will NOT be considered by the Account Debt Monitor (it's already being severed). If a service agreement is stopped, closed, or reactivated, it also will NOT be considered by the Account Debt Monitor (it's already severed).



NOTE:

Multiple collection processes may be kicked off. It's important to be aware that if an account's service agreements reference multiple debt classes, a collection process will be started for each offending debt class.

One collection process per debt class. A given debt class for an account may only have one ongoing collection process at any point in time.

What Happens When A Collection Process Is Started?

When you define collection criteria, you must define the collection process template to use if the criteria are violated. The system uses this template to create the account-specific collection process.

Every service agreement that is part of the offending debt class that has debt older than X days will be linked to the collection process (where X is the debt age on the collection criteria).

Also linked to the collection process will be one or more collection events. These events are typically a series of letters meant to prod the customer (you can also create an event that sends a To Do entry to a user to highlight the offensive debt). You define exactly which letters are generated and when they are generated when you set up the events on your collection process templates.

It's important to note that all of the collection events will be created when the collection process is created. Each of these collection events contains a trigger date. The trigger date of the first event(s) will typically be the current date. The trigger date of the other events will be in the future. Refer to [Calendar vs Work Days](#) for information that describes how the trigger date is set.

A separate process, Activate Collection Events, is responsible for activating collection events whose date is on or before the current date. Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do entry to a user, start a severance processes, etc.)

If adequate payments / credits are recorded in the system, the collection process will be cancelled.

FASTPATH:

For more information about collection process templates, see [Setting Up Collection Process Templates](#). For more information about collection events, see [The Big Picture Of Collection Events](#). For more information about how a collection process is cancelled, see [How Are Collection Processes Cancelled](#).

Experimenting With Alternative Collection Process Templates

The system allows you to determine the efficacy of proposed collection process templates using a small subset of customers before implementing the templates on the entire customer base. We use the term "champion / challenger" to reference this functionality.

We'll use an example to explain. Let's assume your prevailing collection process template for residential customers starts with a "gentle reminder" letter followed 10 days later by a letter threatening collection agency referral if payment is not received. You may want to experiment with the impact of a change to this template. For example, you may want to change the "gentle reminder" to something more assertive and follow this up 5 days later with an even sterner warning. You can use the "champion / challenger" functionality to perform this experiment.

The following points describe how to implement "champion / challenger" functionality:

- Set up a "challenger" collection process template for each template that you want to experiment with.
- Insert a new **Champion/Challenger** option on the Collection Processing [Feature Configuration](#) for every champion template. Each option's value defines:
 - the "champion" collection process template code
 - the "challenger" collection process template code
 - the percentage of the time the system should use the "challenger" template
- Keep in mind that you can only experiment with one challenger template per champion template. For example, let's assume you have two prevailing collection process templates - one for residential customers and another for commercial customers. You can experiment with different challenger templates for the residential and commercial templates. However, you cannot experiment with two different challenger templates for the residential champion template (i.e., a champion template can have 0 or 1 challenger template).

After setting up the above, the [Account Debt Monitor](#) will use the challenger template X% of the time rather than the champion template.

If you are using the Oracle Utilities Analytics product, you can configure analytic zones in innumerable ways to compare the efficacy of the champion versus the challenger. For example,

- You can set up a graph to show the average duration of each type of process.
- You can set up a graph to show the average dollars that were successfully collected.
- You can set up a dimensional scorecard to show how each template performed in different regions (or customer classes or ...).
- Etc (the list is limited by your imagination).

How Are Collection Processes Cancelled?

A collection process may be cancelled via the mechanisms described in this section.

The Collection Process Monitor Can Cancel A Collection Process

The Collection Process Monitor (CPM) is a background process that reviews a collection process when the debt associated with one of its service agreements is reduced. Financial events that can cause service agreement debt to be reduced are:

- The cancellation of a bill segment.
- The creation of a payment segment.
- The creation of an adjustment that credits a service agreement.

The review performed by the CPM occurs as follows:

- **Debt class cancel criteria.** In general, the sum of all debt associated with the collection process's debt class must be less than a given threshold amount for a collection process to be cancelled. If so, the collection process is cancelled.
- Please be aware that, if a [Pay Plan](#) exists for the account and debt class, the customer's debt will be temporarily reduced by the amount of the pay plan's scheduled payments before it is compared to the threshold amount. Please be aware that this temporary reduction will only occur if you have plugged in the appropriate pay plan debt reduction algorithm on the debt class.

NOTE:

The above logic is not "hard coded". The CPM calls the [Collection Process Cancel Criteria Algorithm](#) defined on the debt class that is associated with the collection process. This algorithm will cancel a collection process if the sum of ALL service agreements in the debt class have debt less than a given threshold amount. However, because it's an algorithm, you can introduce whatever cancellation criteria you please.

-
- **Service agreement cancel criteria.** You can optionally introduce a special quirk to the cancellation logic. This quirk is a bit difficult to understand. To understand it, you should recall:
 - All service agreements that are in arrears in a given debt class are linked to the collection process.
 - The collection event called Start Severance creates a severance process for every service agreement that is in arrears on the collection process (the alternative is to [Nominate A Single Service Agreement To Sever](#)).
 - If you use the Start Severance collection event, you would want to remove a service agreement from a collection process when it no longer has intolerable debt (regardless of the state of the debt class's entire debt). You'd want to do this because, if you don't, the system would start a severance process for the paid up service agreement and if it's paid up, you wouldn't want a severance process created for it.
 - To "remove" service agreements from a collection process when they no longer have intolerable debt, you should plug-in a [Service Agreement-Oriented Cancel Criteria Algorithm](#) on your collection process templates. The CPM will call this algorithm if you've plugged it in.

NOTE:

When all service agreements are "removed" from a collection process, the CPM cancels all pending collection events and cancels the collection process.

WARNING:

Checking if individual service agreements should be removed from a collection process is optional (meaning that you don't have to plug one in on the collection process template).

A New Payment Plan Can Cancel A Collection Process

Refer to [Collection Process / Severance Process Cancellation When A Pay Plan Is Created](#) for the details.

NOTE:

Real time cancellation. Please be aware that the system will cancel a collection process real time when a pay plan is created (if the pay plan's scheduled payments are enough to pay-off the customer's outstanding debt).

A User May Cancel A Collection Process At Their Discretion

A user may cancel a collection process at their discretion.

Stopping A Service Agreement May Cancel A Collection Process

The system will "remove" a service agreement from a collection process when it is stopped (i.e., when the service agreement's status becomes Stopped). When the last service agreement is "removed" from the collection process, the collection process will be cancelled.

The Big Picture Of Collection Events

This section describes the various types of collection events and their lifecycle.

How Are Collection Events Created?

Collection events may be created as follows:

- The Account Debt Monitor creates a collection process when an account violates collection criteria. The collection process has one or more collection event(s). The number and type of events is controlled by the collection process template associated with the collection process.
- Collection events are created when a user creates an ad hoc collection process. The number and type of events is controlled by the collection process template defined when the collection process is created.
- An ad hoc collection event may be created and linked to an existing collection process by a user at their discretion.

NOTE:

Bottom line. Most collection events are created by the system when it creates a collection process for delinquent accounts. If you need to create an ad hoc collection event, you can either create a collection process whose template contains the desired event OR link the desired event to an existing collection process.

FASTPATH:

For more information about the creation of events by the Account Debt Monitor refer to [What Happens When A Collection Process Is Started?](#). For more information about creating ad hoc collection processes, refer to [Collection Process Maintenance](#). For more information about creating ad hoc events, refer to [Collection Process - Events](#).

Types Of Collection Events

The following table describes the various types of collection events and what happens when they are completed:

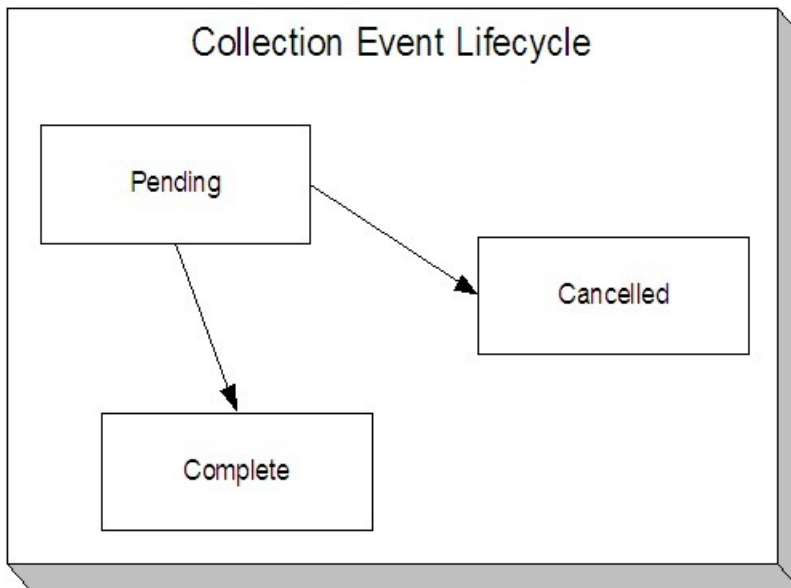
Type Of Collection Event	What Happens In The System
Send Letter	<p>A customer contact is created for every financially responsible person linked to the account. The customer contact causes a letter to be produced.</p> <p>The type of letter is defined on the customer contact's contact type.</p> <p>The recipient of each letter is defined on Account / Person (those persons marked as Receiving Notifications).</p>
Create To Do Entry	A To Do entry is created. Refer to The Big Picture of To Do Entries for more information about To Do entries.
Affect Credit Rating/Cash-Only	An account credit rating demerit record is created. The number of demerits is defined on the collection event type.
Cancel Budget	Every service agreement linked to the account that is on a budget is "removed" from the budget (i.e., the recurring charge amount for the service agreement is set to zero). In addition, a syncing adjustment is issued to cause each SA's current balance to be set equal to their payoff balance (the adjustment type is defined on the SA's SA type). Also, the Budget Plan field on the Account - Budget page is cleared.
Generic Algorithm	The algorithm defined on the event's event type is executed.
Start Severance Process	A severance process is created for every service agreement linked to the collection process. The type of severance process is defined on the SA type table (every service agreement references an SA type).

FASTPATH:

Refer to [Setting Up Collection Event Types](#) for more information.

Collection Event Lifecycle

The following diagram shows the possible lifecycle of a collection event:



Collection events are initially created in the pending state.

When the system sees a pending event with a trigger date on or before the current date, the system executes the event's activity and completes the event.

FASTPATH:

For more information about a collection event's trigger date, see [Collection Event Trigger Date](#).

A pending event will be cancelled automatically by the system when the account's debt no longer violates the collection criteria that sparked the event's collection process. A pending event may also be cancelled by a user at their discretion. Refer to [How Are Collection Processes Cancelled](#) for more information about how the system will cancel a collection process (and its events).

Collection Event Trigger Date

When a collection event is created by the system, its trigger date is set in accordance with your date arithmetic preferences. Refer to [Calendar vs. Work Days](#) for more information.

How Are Collection Events Completed

A background process runs periodically (at least daily) that looks for collection events with a trigger date on or before the current date. For each triggered event, the system executes its activity and then completes it. Refer to [Collection Event Activator](#) for more information.

The Last Collection Event Should Kick Off Severance Process(es)

The last collection event will typically kick off the severance process for every service agreement linked to the collection process. This will only happen if you set up the collection process template accordingly (i.e., the last event type in the process template is the kind that starts a severance process for every service agreement linked to the collection process).

NOTE:

Nominating a service agreement to sever. Many organizations that have multiple services in arrears will NOT sever every service agreement that's in arrears. Rather, they will nominate one service agreement and use it to encourage the customer to pay for the other services. If your organization works this way, then your last collection event should call the [Nominate A Service Agreement To Sever Algorithm](#).

How Are Collection Events Canceled?

Users can cancel a collection event at their discretion. In addition, the system can cancel a collection event when it automatically cancels a collection process. Refer to [How Are Collection Processes Cancelled](#) for the details.

The Big Picture Of Severance Process Cancellation

The topics in this section provide high level information about the cancellation of severance processes.

FASTPATH:

For more information refer to [The Lifecycle Of A Severance Process And Its Events](#).

How Are Severance Processes Cancelled?

A severance process may be cancelled via the mechanisms described in this section.

FASTPATH:

Refer to [What Happens When A Severance Process Is Cancelled?](#) for what happens when a severance process is cancelled.

The Freezing Of Certain Financial Transactions Can Cancel A Severance Process

NOTE:

The system will only cancel a severance process if its severance process template indicates that **Auto Cancel** is allowed. Typically, this switch is set on all severance process templates except for the odd ones that are used to [reconnect service](#).

The system reviews a severance process real-time whenever its service agreement's debt is reduced. Financial events that can cause service agreement debt to be reduced are:

- The cancellation of a bill segment.
 - The creation of a payment segment.
 - The creation of an adjustment that credits a service agreement.
-

NOTE:

Real time cancellation. Unlike collection processes, the system cancels severance processes real time (i.e., there is no background process that monitors severance processes). Why are severance processes canceled real time? Because a severance process may have events that create field activities to sever service. These events need to be canceled the moment the FT is frozen, we can't wait until a background process runs. This means that if a customer pays in person

for a service agreement that is pending severance, the system will cancel the process and its field activities (if any) the moment the payment is entered.

The review takes place as follows:

- **Debt class cancel criteria.** In general, the sum of all debt associated with the severance process's debt class must be less than or equal to a given threshold amount for a severance process to be cancelled. If so, the severance process is cancelled.
- Please be aware that, if a [Pay Plan](#) exists for the account and debt class, the actual debt will be temporarily reduced by the amount of the pay plan's scheduled payments before it is compared to the threshold amount. Note: this temporary reduction will only occur if you have plugged in the appropriate pay plan debt reduction algorithm on the debt class.

NOTE:

The above logic is not "hard coded". The system calls the Severance Process Cancel Criteria Algorithm defined on the [debt class](#) that is associated with the severance process. This algorithm cancels a severance process if the sum of ALL service agreements in the debt class have debt less than or equal to a given threshold amount. However, because it's an algorithm, you can introduce whatever cancellation criteria you please.

- **Service agreement cancel criteria.** You can optionally introduce a special quirk to the cancellation logic. This quirk is a bit difficult to understand. To understand it, you should recall:
 - The collection event called Start Severance creates a severance process for every service agreement that is in arrears. Note: you would only use this type of collection event if you do not [Nominate A Single Service Agreement To Sever](#).
 - If you use the Start Severance collection event, then you would want to cancel a severance process when its service agreement no longer has intolerable debt (regardless of the state of the debt class's entire debt).
- To cancel a severance process when its related service agreement no longer has intolerable debt, you should plug-in a Cancel Criteria Algorithm on your [severance process templates](#). The system will call this algorithm if you've plugged it in.

NOTE:

Manual Creation. A user can create a severance process for an account that does not qualify to be on severance according to the cancel criteria algorithm. For example, perhaps your cancel criteria algorithm cancels a severance process when the account's debt falls below a threshold amount. A user can create a severance process for an account whose debt is already below this threshold. Because cancellation is real time, there is no action that will cause this severance process to be canceled. When a manual severance process is created, the system executes the appropriate cancellation criteria algorithm. If the algorithm indicates that the system would have canceled this severance process, a warning is issued.

A New Payment Plan Can Cancel A Severance Process

Refer to [Collection Process / Severance Process Cancellation When A Pay Plan Is Created](#) for the details.

NOTE:

Real time cancellation. Please be aware that the system will cancel a severance process real time when a pay plan is created that pays off enough debt.

A User May Cancel A Severance Process At Their Discretion

A user may cancel a severance process at their discretion.

Stopping A Service Agreement Will Cancel A Severance Process

The system will cancel a severance process if its service agreement is stopped (i.e., when the service agreement's status becomes Stopped).

What Happens When A Severance Process Is Cancelled?

The following takes place when a severance process is canceled by the system:

- The system cancels all pending severance events and deactivates the severance process.
- If there are any field activities linked to the severance process, an optional plug-in spot defined on the installation record allows you to plug in an algorithm to cancel these field activities.
 - The base package [Severance Process Cancellation Algorithm](#) will cancel all pending field activities that were created as a result of the severance process that are not linked to a dispatched field order.
- If there are any pending field activities left associated with the severance process, it is marked to trigger the creation of a To Do entry to highlight that field activities exist for a canceled severance process. (This happens if you have not plugged in an algorithm to perform the cancellation or if the algorithm detected a condition that prevented cancellation.) To create the To Do entry, you must run the background process [TD-SPRO](#).
- There is an optional plug-in spot defined on the severance process' template. If an algorithm is plugged-in, it is called. The base package algorithm will create a reconnect process if there are completed field activities for a cut for nonpayment severance event associated with the severance process. Refer to [Severance Post Cancellation Algorithm](#) for more information about this algorithm.

The Big Picture Of Severance Events

This section describes the various types of severance events and their lifecycle:

How Are Severance Events Created?

Severance events may be created as follows:

- The process that completes (i.e., executes) collection events creates a severance process when it completes a "start severance process" collection event. The severance process has one or more severance event(s). The number and type of events is controlled by the severance process template associated with the severance process. Refer to [The Collection Event Activator](#) for more information about this process.
- Severance events will be created when a user creates an ad hoc severance process. The number and type of severance events is controlled by the severance process template associated with the severance process.
- An ad hoc severance event may be created and linked to an existing severance process by a user at their discretion.

NOTE:

Bottom line. Most severance events are created by the system when it creates a severance process for delinquent service agreements. If you need to create an ad hoc severance event, you can either create a severance process whose template contains the desired event OR link the desired event to an existing severance process.

FASTPATH:

For more information about creating ad hoc severance processes and events, refer to [How To Perform Common Severance Process Functions](#).

Types Of Severance Events

The following table describes the various types of severance events and what happens when they are completed:

Type Of Severance Event	What Happens In The System
Send Letter	<p>A customer contact is created for every financially responsible person linked to the service agreement's account. It is the customer contact that causes a letter to be produced.</p> <p>The type of letter is defined on the customer contact's contact type.</p> <p>The recipient of each letter is defined on Account / Person (those persons marked as Receiving Notifications).</p>
Create To Do Entry	<p>A To Do entry is created. Refer to The Big Picture of To Do Entries for more information about To Do entries.</p>
Create Field Activities	<p>A field activity is created for each service point associated with the service agreement being severed. The type of activity is defined on the service point's SP type's field activity type profile.</p>
Generic Algorithm	<p>The algorithm defined on the event's event type is executed.</p>
Expire Service Agreement	<p>The service agreement is expired and, if earlier severance events created "cut for non-payment" field activities, these field activities will be used as the basis for stopping service. Refer to Finalizing Pending Stops for how the system use the meter reads on these field activities as the "stop reads" on the service agreement. Note, you can see the field activities that are used to "cut" and "stop" service by viewing the Field Activities grid on Service Agreement - Service Point.</p>
Affect Credit Rating/Cash-Only	<p>An account credit rating demerit record is created. The number of demerits is defined on the collection event type.</p>

FASTPATH:

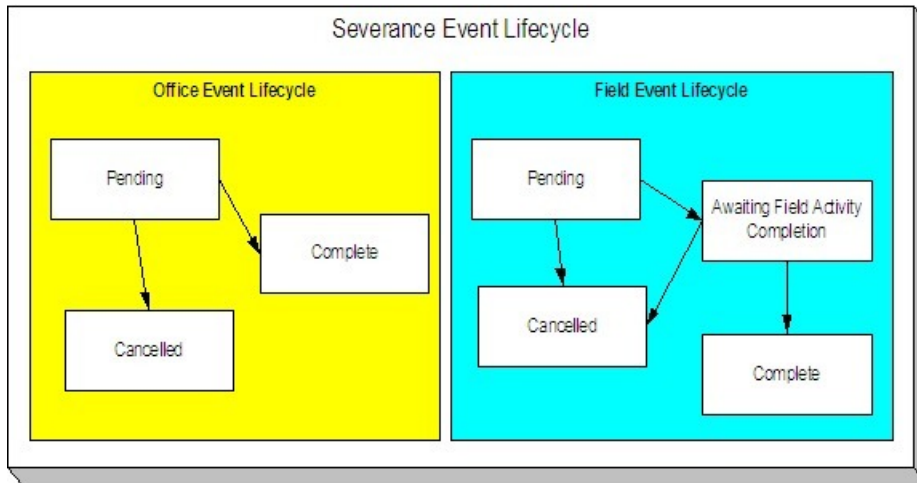
Refer to [Setting Up Severance Event Types](#) for more information.

Field Events Versus Office Events

Severance events are considered either field or office events. Office events are just like collection events in that they don't involve any type of field activity in their completion. Field events, however, are not like collection events because they create one or more field activities. These activities cause these events to have a different lifecycle. Refer to [Severance Event Lifecycle](#) for more information.

Severance Event Lifecycle

The following diagram shows the possible lifecycle of a severance event:



The following points explain the lifecycle of severance events of the office variety:

- Office events are initially created in the pending state.
- A pending office event becomes complete when the system sees that its trigger date is on or before the current date. At this time, the system executes the event's activity (e.g., create a letter, create To Do entry).

FASTPATH:

For more information about a severance event's trigger date, see [Severance Event Dependencies & Trigger Date](#).

- A pending office event will be cancelled automatically by the system when the debt associated with the severance process's service agreement is sufficiently reduced. A pending office event may also be cancelled by a user at their discretion. Refer to [How Are Severance Processes Cancelled](#) for more information about how the system will cancel a severance process (and its events).

The following points explain the lifecycle of severance events of the field variety:

- Field events are initially created in the pending state.
- A pending field event becomes awaiting field activity completion when the system sees that its trigger date is on or before the current date. At this time, the system creates the field activities associated with the given event (e.g., disconnection warning, disconnect for nonpayment, etc.).

FASTPATH:

For more information about a severance event's trigger date, refer to [Severance Event Dependencies & Trigger Date](#). For more information about the field activities that are created for a field severance event, refer to [Field Events And Their Activities](#).

- An awaiting field activity completion field event becomes complete when the system sees that its field activities are all complete or cancelled.
- A pending field event will be cancelled automatically by the system when the service agreement associated with the severance event's severance process has sufficient credits. A pending field event may also be cancelled by a user at their discretion.
- An awaiting field activity completion field event will be cancelled automatically by the system when the service agreement associated with the severance event's severance process has sufficient credits if the field activity has not been dispatched (refer to [Designing Your Reconnection Procedures](#) for information on how the system handles the situation if the field activity is completed or dispatched). An awaiting field activity completion event may also be cancelled by a user at their discretion.

Severance Event Dependencies & Trigger Date

When a severance event is created by the system, its trigger date cannot be set. This is because, unlike collection events, the trigger date on severance events can only be set when ALL of the preceding severance events on which it depends are complete. An example will help explain why this design is necessary. Consider the following example that shows a standard severance process and its events:

Event Number	Severance Event	Dependent On Event(s)	Trigger Date Set To X Days After Completion Of Preceding Events
10	Field activity - Disconnect for non-payment warning	N/A - first event	0
20	Field activity - Disconnect for non-payment	10	2
30	Create To Do entry	20	0
40	Send letter to customer	20	0
50	Expire service agreement	20	10

This severance process is meant to execute as follows:

- On the first day, generate a 48-hour warning of impending disconnection. This is a field event as most organizations deliver this warning in person.
- After 2 days (i.e., 48 hours) have passed, create a field activity to disconnect for non-payment.
- When this is completed, generate a To Do entry to let a CSR know about the cutoff. Also, send a letter to the customer.
- If after 10 days from the cutoff, we still don't have payment, expire the service agreement.

As can be seen from the above example, the later events are dependent on the completion of the field activities in the earlier events. This means that when you set up a severance event, you must indicate the events on which it depends (and the number of days after their completion that the event should be triggered).

NOTE:

Bottom line. The system sets the trigger date on severance events when it detects that all of its dependent events are complete (this is the responsibility of the SED background process). Refer to [Calendar vs Work Days](#) for a description of your choices in respect of how the trigger date is calculated.

Field Events And Their Activities

When the system is told to start a severance event that creates field activities (e.g., disconnect for non-payment), it will generate a field activity for every service point linked to the service agreement.

The question is, where does it get the field activity type associated with the field activities? The answer is explained below:

- Every service point has an SP type.
- Every SP type references a field activity type profile.
- A field activity type profile contains a matrix defining activity types to generate under various situations. Some of these situations are those associated with severance.

FASTPATH:

For more information, refer to [Setting Up Field Activity Type Profiles](#).

Severance Event Activation & Completion

A background process runs periodically (at least daily) that looks for severance events with a trigger date on or before the current date. This process executes the activity associated with each event. If the event is an office event, the event then becomes complete. If the event is a field event, the event becomes "awaiting field activity completion" until the field activities are complete. At this time, the severance event is completed. Refer to [Severance Event Activator](#) for more information.

How Are Severance Events Canceled?

Users can cancel a severance event at their discretion. In addition, the system can cancel a severance event when it automatically cancels a severance process. Refer to [How Are Severance Processes Cancelled](#) for the details.

The Big Picture Of Write Off Processing

Before you financially write-off debt, most companies go to some effort to collect the past due funds. You control exactly what happens by setting up the various write-off control tables. The topics in this section provide background information that will help you understand how the information in these control tables is used.

How Is Debt Financially Written-Off?

Before debt can be written-off, a write-off service agreement must exist for the account. Why? Because when you write-off a normal service agreement's debt, you are actually transferring its debt to a write-off service agreement.

A write-off service agreement is just like other service agreements in that:

- It holds debt.
- When a payment is received, the service agreement's debt is reduced.

Debt is transferred to a write-off service agreement (WO SA) from the customer's uncollectable service agreements (SAs). The following points highlight important characteristics about the uncollectable SAs and the WO SA:

- The WO SA and the uncollectable SAs should be linked to the same account (note: this isn't a strict rule, it just makes sense because an account's written off funds should be linked to the account).
- Debt may be transferred to a WO SA from any type of service agreement regardless of debt class, i.e., a WO SA can contain debt that originated in any debt class.
- When you transfer debt from the uncollectable SAs to the WO SA, the debt is removed from the uncollectable SAs (and their status becomes closed - assuming their balance becomes zero).
- If you use the system's automated write-off processing, the system will create WO SAs for you. The system's automated write-off processing can write-off revenue in a different manner than is used to write-off liabilities. Refer to [The Ramifications of Write Offs in the General Ledger](#) for more information.
- WO SAs are immune from the account debt monitor (assuming their debt class is marked as not being subject to collection activities).
- WO SAs are not billed (assuming their SA type is marked as being not billable).

- WO SAs start their life with a non-zero payoff and current balances (i.e., they have debt when first started). This debt is transferred from the normal service agreement(s) whose uncollectable debt necessitated the creation of the WO SA.
- If the customer pays off the write-off debt, the WO SA remains active in case you ever need to write-off debt in the future. If you don't like the WO SA remaining active after it's paid off, you can indicate on the WO SA's SA Type that it is a "one time charge", this will cause the WO SA to be automatically closed when it's paid off.
- You can transfer additional uncollectable debt to the WO SA.

NOTE:

Bankruptcy write-offs. If you have to write-off debt because a customer declares bankruptcy, everything stated above is true. The only thing you have to do is use a different SA type for bankruptcy write-offs as compared to "normal" write-offs. On the bankruptcy write-off SA type, simply leave the payment segment type blank - this way the system will never distribute a payment to the bankrupt debt (because bankrupt debt is legally uncollectable).

The Ramifications of Write Offs in the General Ledger

WARNING:

If you practice cash accounting, refer to [Cash Accounting and Write-Offs](#).

When you write-off unpaid debt, you shouldn't book it all to a write-off expense account. Why? Because the debt that you're writing off typically contains both revenue and liabilities. At write-off time, you typically want to:

- Book the written off revenue to a write-off expense account, and
- Reduce the liabilities (you don't owe the liability if you don't get paid).

Consider the following example of a simple electric service agreement with two financial transactions:

Event	GL Accounting
Customer is billed	A/R 1000
	Revenue <900>
	State Tax Payable - Taxing State - California <80>
	City Tax Payable - Taxing City - San Francisco <20>
Customer is levied a late payment charge	A/R 50
	Late Payment Revenue <50>

After these two financial transactions are booked, the customer has debt of \$1050. Of this \$1,050; \$950 is revenue and \$100 is liability (money you owe the taxing authorities).

If the customer doesn't pay, you will eventually have to write-off this debt. Most organizations would issue the following types of financial transactions to do this:

Event	GL Accounting
Write-off the bill	Write-off Expense 900
	State Tax Payable - Taxing State - California 80
	City Tax Payable - Taxing City - San Francisco 20
	A/R <1000>
Write-off the late payment charge	Write-off Expense 50

Notice in the above transactions, the two separate revenue accounts are written off by booking to an expense account. However, the liability accounts are reversed. Why is revenue treated differently from liabilities at write-off time? There's a good reason for it (if you're an accountant), for the time being, just accept that this is how it works.

And finally, we need to worry about what happens if the customer eventually pays off his written off debt. If this happens, most organizations would pay off the write-off first, and, if there was still money left, they'd reimburse the taxing authorities. If we assume the customer pays off the entire written off debt, the following financial transactions would be issued:

Event	GL Accounting
Pay off the written off debt	Cash 900
	Write-off Expense <900>
Reinstate the liabilities	Cash 100
	State Tax Payable - Taxing State - California <80>
	City Tax Payable - Taxing City - San Francisco <20>

While the reinstatement of liabilities at payment time is possible in the system, the ramifications of doing such make this approach impracticable (the ramifications are a) if the check bounces, we would not be able to reduce the liabilities, and b) if there was a partial payment of the liabilities, the remaining unpaid amount could get written down). Therefore, when a write-off is paid the following financial transactions should be issued:

Event	GL Accounting
Pay off the written off debt	Cash 900
	Write-off Expense <900>
Reinstate the liabilities	Cash 100
	Reinstated liabilities <100>

Notice that rather than reinstating the individual liabilities, we simply reinstate all liabilities into a single account. This means your accountants will have to distribute this money to the appropriate liabilities manually.

So, how do we achieve the above in the system? This explanation is a little complicated, but it'll make sense if you keep the above financial transactions in mind:

- First of all, you'll need two different SA types - one to hold the written off revenue and another to hold the reduced liabilities.
 - On the SA type that holds written off revenue, indicate that it is not billable, indicate that it cannot have excess credits, and give it a high payment distribution priority. The distribution code on this SA type should reference your Write-off Expense account.
 - On the SA type that holds the reduced liabilities, indicate that it is not billable, indicate that it cannot have excess credits, and give it a high payment distribution priority. The distribution code on this SA type should reference a the "reinstated liabilities" GL account.

Next, you need to understand how the system's standard write-off logic works:

- The system accumulates the distribution codes from GL details associated with recent financial transactions linked to the service agreement being written-off.
- When the system has accumulated enough distribution codes (i.e., where the amount associated with the distribution code equals or exceeds the amount to write off), the debt will be transferred to a new or existing write-off service agreement(s). The number and type of service agreements to which the bad debt is transferred is defined on the distribution codes. Refer to [Setting Up Distribution Codes](#) for how to define the type of write-off service agreement

associated with a distribution code. In our example, we'd need the two SA types described above - one for the revenue accounts, the other for the liability accounts.

- At write-off time, for those distribution codes associated with revenue, the system will create a transfer adjustment from the normal service agreement to the write-off revenue service agreement. This will reduce (credit) the receivable on the normal service agreement and increase (debit) the expense account defined on the write-off revenue service agreement.
- However, if we do the above for the distribution codes associated with liabilities, we have a problem. The problem is a bit hard to explain unless you understand tax accounting, but it basically comes down to this - if we simply transfer the portion of the receivable balance associated with the liabilities to the write-off liability SA, we will always be debiting the distribution code defined on the SA type. This isn't correct because we really want to debit the liability account (and reference the characteristic type and value from the original credit) when we reduce the liability. So how do we do this? For those distribution codes associated with liabilities, you need to indicate that you want to override the distribution code on the "transfer to" side of the transfer adjustment with the distribution code / characteristic type / characteristic value that was originally booked. Refer to [Setting Up Distribution Codes](#) for how to indicate you want to override the distribution code at write-off time. If you do the above, then at write-off time the transfer adjustment will reduce (credit) the receivable on the normal service agreement and increase (debit) the original liability accounts from the original financial transactions.

If you followed the above, you'll see that we now have everything debited and credited appropriately. And, if a payment materializes for the written off debt, we will simply debit cash and credit the distribution code on the respective SA (either Write Off Expense or Reinstated Liabilities).

NOTE:

Batch and real-time write-offs may use the above processing. The above logic is executed real time when a user writes off debt using the [write-off transaction](#) (assuming the base package [write off algorithm](#) is plugged into the account's [customer class](#)). The above logic is executed in batch when a write-off event that references a Write Off Using Distribution Codes [event type](#) is executed. Write-off events are described in detail below.

Automated versus Manual Write Offs

The system will automatically create write-off SAs and transfer uncollectable debt to them during the automated write-off processing described below.

If necessary, you can write-off debt outside of the automated write-off process using either of the following methods:

- You can transfer bad debt from any service agreement to a write-off service agreement using a transfer adjustment.
- You can use the [write-off transaction](#) to write-off debt real-time. When this transaction is used, the system executes the logic embedded in the Write Off Method algorithm that's plugged in on the account's [customer class](#).

How Does The Write-Off Monitor Work?

This section describes how the [Write Off Monitor](#) uses your write-off criteria and write-off process templates to collect overdue debt.

Different Write-Off Criteria For Different Customers And Different Debt

Consider the following:

- You probably have different write-off criteria for different customer segments. For example, customers with large bills probably have strict criteria, whereas you're probably more lenient with small customers (or vice versa). You differentiate your customers in respect of the collection process via a **collection class code on the customers' accounts**.

An account's initial collection class is defaulted from its customer class. You may override an account's collection class at will.

- You probably have different write-off criteria for different classes of debt. For example, if a customer has both regulated and unregulated debt, you probably have commission-imposed criteria to be applied to the regulated debt, but you have control over how to write-off unregulated debt. You differentiate your debt in respect of the collection process via a write-off **debt class on the customers' service agreements** (note the write-off debt class is actually defined on the SA type and every service agreement has a SA type).

NOTE:

Write Off Debt Class vs. Regular Debt Class. It's important to be aware that a SA type references both a regular debt class and a write-off debt class. The regular debt class controls the collection criteria applied against an account's service agreements. The regular debt class is also used to segregate an account's outstanding balance on several queries in the system. The write-off debt class controls the write-off criteria applied against an account's stopped service agreements. The reason the system supports two different debt classes is because you may categorize your service agreements differently when you try to collect overdue debt versus when you write-off debt.

Given the above, you should understand that different write-off criteria will exist for every combination of collection class and write-off debt class. If you're confused, then consider the following matrix:

SA's Write-Off Debt Class	Account's Collection Class:	
	Commercial Customer	Residential Customer
Regulated	N/A - there is no regulated, commercial customer debt.	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. Attempt to transfer debt to another active service agreement linked to the account. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment. If the debt is <= \$-1, create an A/P adjustment to refund the credit to the customer. If debt remains, create the default write-off process for regulated debt.
Unregulated	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. Attempt to transfer debt to another active service agreement linked to the account. If the debt / credit is < \$10 and > \$-10, write down the debt using a write-down adjustment. If the debt / credit is <= \$-10, create an A/P adjustment to refund the credit to the customer. If debt still remains: Highest priority: If customer has a non-cash deposit, create the non-cash deposit write-off process.	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. Attempt to transfer debt to another active service agreement linked to the account. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment. If the debt is <= \$-1, create an A/P adjustment to refund the credit to the customer. If debt remains, create the default write-off process for unregulated residential debt.

Otherwise, create the default write-off process
for unregulated commercial debt.

Notice that each cell in the matrix has the same pattern:

- The system first attempts to reduce the SA's current and payoff balances to zero using the following methods (assuming you have set up the write-off control appropriately):
 - Sync the current balance with the payoff balance. If the SA's payoff balance is zero, this will cause the current balance to become zero and therefore close the SA.
 - If there's a debit balance, transfer the debt to any pending start or active SA in the same write-off debt class.
 - If there's a credit balance, transfer the debt to any non-closed / non-cancelled SA in the same write-off debt class
 - If the remaining debit / credit balance is within a user-defined tolerance (this is defined on the respective algorithm on the write-off control), create an adjustment to write-down the small balance.
 - If a credit balance remains, create an A/P adjustment to refund the balance with a check (the adjustment type is defined on the respective algorithm on the write-off control).
- All of the above points will cause the SA to close. If debt remains, the system starts some type of write-off process. The type of process is dependent on the respective criteria. What differentiates one write-off criteria from another is its priority. The higher priority criteria will be compared first. If the customer / debt meets the criteria, the write-off process is initiated; no further comparisons are performed.

FASTPATH:

For more information about maintaining this matrix, refer to [Setting Up Write-off Control](#).

When Is Debt Monitored For Write Off Purposes?

The write-off monitor only reviews a service agreement when the following conditions are true:

- The service agreement is stopped and reactivated.
- If the service agreement is a "billable charge" SA (as identified on its SA type), all of its billable charges must appear on a bill segment AND the bill segment's bill's due date plus grace period must be on or before the business date.
- If the service agreement is not a "billable charge" SA AND it is billable (as identified on its SA type), the SA must have a closing bill segment (i.e., it must be final billed) and the bill segment's bill's due date plus grace period must be on or before the business date.
- If the service agreement is a sub SA, its master SA must abide by the above conditions.
- If the service agreement is not billable, it is possible that adjustments, which affect the SA's debt, exist. The write-off monitor will only review a non-billable SA if all FTs for this SA that have been marked to include on a bill have been swept onto a bill and the bill for any of these FTs has a bill due date plus grace period on or before the business date.

NOTE:

Postponing write-off processing. You can prevent the write-off process from processing an eligible service agreement by populating the account's C&C Postpone Date with a future date.

Attempt To Close The SA Before Creating A Write Off Process

Before the write-off monitor creates a write-off process for a stopped and reactivated service agreement, it attempts to reduce the service agreement's debt to zero using all of the following methods:

- If the account has active service agreements, it will transfer the finalized debt to a pending start or active service agreement.
- If the debt or credit amount on the service agreement is small, the system will generate an adjustment to 'write it down' (or up in the case of a small credit).
- If the service agreement has a large credit amount, the system will generate an A/P adjustment (resulting in a check being sent to the customer).

NOTE:

Plug-in algorithms do the work. Algorithms that are plugged-in on the [write-off control](#) responsible for managing the service agreement's debt actually perform the above effort. You can customize these algorithms to behave exactly how your collections staff desires.

If the algorithms responsible for the above effort are successful in reducing the service agreement's debt to zero, then the service agreement closes and will not be subject to write-off processing. If the above algorithms don't result in the service agreement's debt being reduced to zero, a write-off process will be started (as describe below).

What Happens When A Write-Off Process Is Started?

When you define write-off criteria, you must define the write-off process template to use if the criteria are violated. The system uses this template to create the account-specific write-off process.

Every stopped or reactivated service agreement that is part of the offending write-off debt class will be linked to the write-off process.

Also linked to the write-off process will be one or more write-off events. These events are meant to prod the customer. You define the types of events and when they are generated when you set up your write-off process templates.

It's important to note that all of the write-off events will be created when the write-off process is created. Each of these write-off events contains a trigger date. The trigger date of the first event(s) will typically be the current date. The trigger date of the other events will be in the future. Refer to [Calendar vs Work Days](#) for a description of how the trigger date is calculated.

A separate process, Activate Write-off events, is responsible for activating write-off events whose date is on or before the current date. Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do to an operator, refer debt to a collection agency, etc.)

NOTE:

Multiple write-off processes may be kicked off. It's important to be aware that if an account's service agreements reference multiple write off debt classes, a write-off process will be started for each offending write off debt class that has stopped or reactivated service agreements.

FASTPATH:

For more information about write-off process templates, see [Setting Up Write Off Process Templates](#). For more information about write-off events, see [The Big Picture Of Write-off Events](#). For more information about how a write-off process is cancelled, see [How Does A Write-Off Process Get Cancelled?](#).

How Does A Write-Off Process Get Cancelled?

The system "removes" a service agreement from a write-off process when its status becomes closed (i.e., when its balance is zero). When all service agreements are removed, the system cancels all pending write-off events and deactivates the write-

off process. When the write-off process is deactivated, all collection agency referrals associated with the write-off process are cancelled.

NOTE:

Removing closed service agreements from a write-off process. Service agreements aren't actually removed from the process. Rather, they are inactivated so a proper audit exists.

How Do Collection Agency Referrals Work?

The following points describe how collection agency referrals work.

- A write-off process has one or more events. One type of event causes overdue debt to be referred to a collection agency.
- When a referral write-off event is activated, the system marks the event for processing by the event's Collection Agency Referral Algorithm (refer to [Setting Up Write Off Event Types](#) for more information).
- The next time the Collection Agency Referral process executes (the frequency is dependent on your background process schedule), it will refer the process' debt to a collection agency. The specific agency to which the debt is referred is controlled by the event type's Collection Agency Referral Algorithm. The sample algorithm supplied with the system simply refers debt to the collection agency with the least amount of referred debt. If you prefer different logic, you must write your own algorithm.
- Regardless of the manner in which a collection agency is selected for an account's debt, the referral involves the creation of a collection agency referral history record. Refer to [Collection Referral](#) for more information.
- A collection agency referral history record is linked to an account. It contains the amount of debt referred to the collection agency. It is the creation of this record that, in turn, triggers the interface of information to the collection agency. The method used to interface the information to the agency is defined on the collection agency's record. Refer to [Setting Up Collection Agencies](#) for more information.
- If the collection agency is successful in obtaining the funds, simply add a payment. If the payment causes the SA's balance to become zero, the system will automatically close the service agreement. When the system closes a service agreement, it is "removed" from the write-off process. When a write-off process no longer contains active service agreements, the system cancels the write-off process. When a write off process is cancelled, all collection agency referrals are automatically cancelled.
- Collection agency referrals get cancelled by the creation of a new collection agency referral history record (with a type of cancel). This record will be interfaced to the agency in the same manner used to interface a new referral (see above).
- If the collection agency is not successful in obtaining your funds after a given amount of time, you probably want to cancel the referral and write-off the debt. The cancellation of the referral will happen automatically if you design your write-off process to generate a collection agency cancellation X days after the referral. Refer to [Setting Up Write Off Process Templates](#) for how to do this. You can cancel a referral manually by simply creating a new collection agency referral history record (with a type of cancel).

FASTPATH:

When you enable the Control Central alert algorithm, [C1-COLL-REF](#), an alert displays when an account has an active collection agency referral. This algorithm is plugged-in on the [installation record](#).

The Big Picture Of Write-off Events

This section describes the various types of write-off events and their lifecycle.

How Are Write-off Events Created?

Write-off events may be created as follows:

- The Write-Off Monitor creates a write-off process when an account has unpaid, final billed service agreements. The write-off process has one or more write-off event(s). Refer to [How Does The Write-Off Monitor Work?](#) for more information about how the system creates write-off processes and their events.
- Write-off events are created when an operator creates an ad hoc write-off process. The number and type of events is controlled by the write-off process template defined when the write-off process is created.
- An ad hoc write-off event may be created and linked to an existing write-off process by an operator at their discretion.

NOTE:

Bottom line. Most write-off events are created by the system when it creates a write-off process for unpaid, finalized service agreements. If you need to create an ad hoc write-off event, you can either create a write-off process using a template that contains the desired event OR link the desired event to an existing write-off process.

FASTPATH:

For more information about creating ad hoc write-off processes and events, refer to [How To Perform Common Write-off Maintenance Functions](#).

Types Of Write-off Events

The following table describes the various types of write-off events and what happens when they are completed:

Type Of Write-off Event	What Happens In The System
Affect Credit Rating/Cash-Only	An account credit rating demerit record is created. The number of demerits is defined on the write-off event type.
Cancel Agency Referral	All collection agency referrals associated with the write-off process will be cancelled.
Refer to Agency	The debt associated with the SAs linked to the write-off process will be referred to a collection agency.
Send Letter	A customer contact is created for every financially responsible person linked to the account. The customer contact causes a letter to be produced. The type of letter is defined on the customer contact type control table. The recipient of each letter is defined on Account / Person (those persons marked as Receiving Notifications).
Send To Do	A To Do entry is created. Refer to The Big Picture of To Do Entries for more information about To Do entries.
Write Off using Distrib Code	This type of event is used to write-off bad debt in accordance with the distribution codes associated with the financial transactions that caused the debt in the first place. You'd use this method for example if you want to write-off revenue differently than you write-off liabilities. The system accumulates the distribution codes from GL details associated with recent financial transactions linked to each write-

off service agreement. When the system has accumulated enough distribution codes (i.e., where the amount associated with the distribution code equals or exceeds the amount to write off), the debt will be transferred to a new or existing write-off service agreement. The type of service agreements to which the debt is transferred is defined on the distribution codes.

Write Off using SA Type

The service agreements linked to the process will be written-off by transferring their debt to a new or existing write-off service agreement. Note: the SA type of the write-off service agreement is defined on the algorithm defined on events of this type.

Generic Algorithm

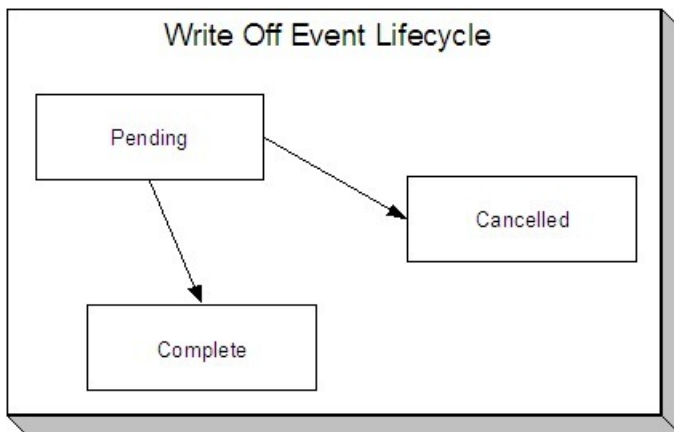
The system calls the algorithm defined on the write-off event type. This type of event is used when what you need to do isn't handled by one of the above event types.

FASTPATH:

Refer to [Setting Up Write Off Event Types](#) for more information.

Write-off Event Lifecycle

The following diagram shows the possible lifecycle of a write-off event:



Write-off events are initially created in the pending state.

When the system sees a pending event with a trigger date on or before the current date, the system executes the event's activity and completes the event.

FASTPATH:

For more information about a write-off event's trigger date, see [Write-off Event Trigger Date](#).

A pending event will be cancelled automatically by the system when the service agreements linked to the process are all closed (i.e., they no longer have debt - either because it was paid or transferred to a write-off service agreement). A pending event may be cancelled by an operator at their discretion.

Write-off Event Trigger Date

When a write-off event is created by the system, its trigger date is set in accordance with your date arithmetic preferences. Refer to [Calendar vs. Work Days](#) for more information.

How Are Write-off Events Completed

A background process runs periodically (at least daily) that looks for write-off events with a trigger date on or before the current date. For each triggered event, the system executes its activity and then completes it. Refer to [Write Off Event Activator](#) for more information.

The Last Write-off Event Should Write Off All Debt Associated With All SAs

The last write-off event will typically transfer all debt from the service agreement's linked to the process to a write off service agreement (linked to the account). This will only happen if you set up the write-off process template accordingly (i.e., the last event in the write-off process template is the kind that writes off debt for every service agreement linked to the write-off process).

How Are Write-off Events Canceled?

The system removes a service agreement from a write-off process when its status becomes closed (i.e., when its balance is zero). Be aware that any type of financial event could cause an SA's balance to fall to zero (e.g., the creation of an adjustment, the application or cancellation of a payment, the cancellation of a bill, ...). When all service agreements are removed, the system cancels all pending write-off events and deactivates the write-off process.

NOTE:

Real time cancellation. Unlike collection processes, the system cancels write-off processes real time when the service agreement becomes closed (i.e., there is no background process that monitors write-off processes).

Besides the automated cancellation process, an operator may cancel a write-off event at will.

NOTE:

Removing closed service agreements from a write-off process. Service agreements aren't actually removed from the process. Rather, they are inactivated so a proper audit exists.

Calendar vs. Work Days

When you set up your collection, severance and write-off process templates, you supply information that controls how the system determines the trigger date of each event in the related process. There are two different mechanisms for doing this:

- When you set up your severance process templates, you must define the number of days between each event. For example, the second event (send cutoff warning) may need to be triggered 7 days after the first event (send reminder letter).
- When you set up your collection and write-off process templates, you must define the number of days after the start of the process when each event should be triggered. For example, the second event (send cutoff warning) may need to be triggered 7 days after the start of the collection process.

The system uses this information in conjunction with the account's division's work calendar when it allocates a trigger date to the various collection, severance, and write-off events in your processes. The system offers you the following choices in respect of how it calculates an event's trigger date:

- You can indicate that the trigger date should be set to the next possible workday. For example, if you indicate that the second event is triggered 7 days after the first event, the system will add 7 days to the first event's completion date. It then checks if this is a workday (and not a holiday), if so, this is the trigger date of the event; if not, it assigns the trigger date to the next workday.
- You can indicate that the trigger date should be calculated by counting workdays. For example, if you indicate that the second event is triggered 7 days after the first event, the system will count 7 workdays (using the account's division's work calendar), and set the trigger date accordingly.

You must define which of the above methods is used in the following processes:

- Account Debt Monitor (ADM and ADM2). Refer to [The Account Debt Monitor](#) for more information.
- Collection Event Trigger (CET). Refer to [The Collection Event Activator](#) for more information.
- Severance Event Set Trigger Date (SED). Refer to [Set Trigger Date](#) for more information.
- Write-off Monitor (WPM). Refer to [The Write Off Monitor](#) for more information.

The Big Picture Of Payment Arrangements and Pay Plans

The topics in this section describe two different mechanisms that allow a customer to payoff overdue debt in installments.

The Big Picture Of Pay Arrangements

A payment arrangement is an agreement with a customer to payoff severely overdue debt in **billed** installments. Bills sent to customers with payment arrangements contain charges for both their current services and their payment arrangement installment amount.

NOTE:

Nomenclature. Some people refer to payment arrangements as "current bill plus" agreements because the customer's bills contain charges for both their current debt plus their installment amount. After the customer has paid off their overdue debt, the customer's bill only contains charges for their current debt.

The topics in this section describe how to set up a payment arrangement and how the system monitors the ongoing arrangements.

Creating Payment Arrangements

When you create a payment arrangement, you are actually creating a service agreement. This service agreement is just like other service agreements in that:

- It holds debt.
- It is periodically billed.
- When a payment is received, the service agreement's debt is reduced.
- If the service agreement becomes delinquent, a collection process is initiated to collect the overdue debt.

Debt is transferred to a payment arrangement service agreement (PA SA) from the customer's delinquent service agreements (SAs) at the inception of the payment arrangement.

When you transfer delinquent debt from the delinquent SAs to the PA SA, the debt is removed from the delinquent SAs. If you transfer all debt from the delinquent SAs, the customer will no longer be in arrears in a given debt class (and if the customer is no longer in arrears, active collection and severance processes will be cancelled).

NOTE:

Use the Payment Arrangement Transaction. You could do the above functions by adding a new service agreement and creating transfer adjustments. However, this is tedious. Rather, use the [Payment Arrangement](#) transaction. This transaction creates the PA SA, transfers debt to it, and sets up the installment amount. This transaction is also used if you need to break or cancel the payment arrangement.

Installment, Payoff and Current Amounts

WARNING:

If you do not understand the difference between payoff balance and current balance, refer to [Current Amount versus Payoff Amount](#).

When you set up a payment arrangement service agreement (PA SA), you transfer delinquent debt to the PA SA using transfer adjustments. After moneys are transferred, the system sets the PA SA's current balance to zero. At this point, neither the original service agreements nor the PA SA have delinquent debt. If the customer neglects to pay their payment arrangement, the PA SA will fall into arrears and a collection process will ensue. If the customer neglects to pay their previously delinquent SAs, they will again fall into arrears and a collection process will ensue.

PA SAs start their life with a non-zero payoff balance (i.e., they have debt when first started). This debt is transferred from the normal service agreement(s) whose outstanding debt necessitated the creation of the PA SA.

The installment amount that the customer is billed is determined by the number of installments used to payoff the debt. For example, if the customer owes \$500 on their electric and water service agreements and they want to pay this off in 10 installments, you'd set up the installment amount to be \$50. The installment amount is saved on the PA SA's recurring charge amount. If the customer again falls into arrears on their normal service agreements, you can transfer additional delinquent debt to the PA SA. You can also change the installment amount as needed.

A PA SA's payoff balance typically differs from its current balance. The payoff balance is the amount of debt remaining to be paid off under the terms of the payment arrangement. The current balance is the installment amount that has been billed but not paid. For example, a customer who is paying off \$500 with 10 installments of \$50 would have an initial payoff balance of \$500 and a current balance of \$0. After the first bill, the PA SA would still have a payoff balance of \$500, but its current balance would be \$50. When the customer pays, the PA SA's payoff balance would fall to \$450 and its current balance would return to \$0.

The following table contains a financial example of a customer who sets up a payment arrangement to payoff \$1,000 of debt in \$10 installments.

Event	Normal SA's GL Accounting	PA SA's GL Accounting	Normal SA's Current Balance	Normal SA's Payoff Balance	PA SA's Current Balance	PA SA's Payoff Balance
Prior to creation of payment arrangement	N/A	N/A	1000	1000	N/A	N/A
Transfer debt from normal SA(s) to PA SA	Xfer 1000 A/R <1000>	PA A/R 1000 Xfer <1000>	0	0	1000	1000
Set current balance to zero on PA SA	N/A	N/A	0	0	0	1000

Customer is billed (\$50 for new debt and \$10 of payment arrangement debt)	A/R 50 Revenue <50>	N/A	50	50	10	1000
Customer pays \$60	Cash 50 A/R <50>	Cash 10 PA A/R <10>	0	0	0	990

When the customer pays off the payment arrangement debt, the system automatically closes the PA SA after it final bills (assuming the PA SA's SA type references a bill segment type that has a bill segment creation algorithm of Recurring Charge With Auto Stop).

Monitoring Payment Arrangements

The PA SA should belong to its own debt class (let's call it Payment Arrangement Debt) so that you can have stricter collection criteria for payment arrangement debt (as compared to normal SAs). Because there will be a new debt class, there will be a unique collection class control (CCC) for payment arrangements. This CCC will have debt criteria associated with payment arrangement debt. If these criteria are violated, we will kick off a collection process that should have 1 collection event - Start Severance.

The severance process template for PA SAs will have 1 severance event that calls the Break Payment Arrangement Event algorithm. This algorithm does the following:

- Cancels ALL adjustments that were used to transfer the debt to the payment arrangement (identified by the XFER adjustment type on the PA SA's SA type). When these are cancelled, the original arrearage will be reinstated under the original SAs - this debt should be rather old by this point.
- Syncs up current balance with payoff balance on the PA SA.
- Makes the PA SA pending stop (SA activation will stop the SA when it next runs).
- If there is a credit left on the PA SA (because payments were made against the arrangement), the credit will be distributed amongst the account's debt using the standard distribution algorithm. Because the payment arrangement debt that was reinstated should be rather old, it should get relieved first. This relief will occur via transfer adjustments from the PA SA to the original SAs.
- If there is a debit left (e.g., because LPC were issued or some other type of adjustment was created by an operator), the debt will be transferred back to one of the SAs from which the arrangement was originally created.
- Inserts a characteristic under the PA SA to indicate that it has been broken (we need this for the account debt monitor (ADM) a few steps down).
- Inserts a row on the account debt monitor trigger. This trigger will cause the account to be reviewed by the ADM when the ADM next runs.

NOTE:

The PA SA must final bill before it closes. The PA SA will only close after the PA SA is final billed. This is OK as it won't have any money left on it.

When the ADM next runs, it will analyze the account's reinstated debt. We recommend creating a new override collection criteria for the normal debt class that will return a value of true if the account has a closed payment arrangement that has been broken in the last X days (where X is a parameter of the override collection criteria's algorithm). If this algorithm returns a true, kick off a unique collection process template (that has nasty events). A sample algorithm of this type is supplied in the base package - COLL COND PA.

To complete this discussion, we have to worry about the situation when the final bill of a payment arrangement goes unpaid. In this situation, the payment arrangement is stopped and will therefore not be processed by the ADM. In this case, the write off monitor will process the PA SA after its final bill's due date and a write-off process will start. This write off process will have a single event that calls the Break Payment Arrangement algorithm (described above). After the FT's are issued in this event, the SA will close (because it's been final billed and it's balance will go to zero).

The Big Picture Of Pay Plans

A pay plan (PP) is an agreement with a customer to make payments on specific dates. Pay plans differ from payment arrangements in that pay plans have user-defined scheduled payment dates, which are independent from the customer's billing dates. In other words, payment arrangements appear on the customer's bills, pay plan scheduled payments do not.

If a customer is in arrears and you want to receive payments on specific dates (as opposed to with the customer's regular bills), you would set up a pay plan and define the dates on which you expect the payments.

The topics in this section describe how pay plans work.

A Pay Plan Has One Or More Scheduled Payments

When you create a pay plan for an account, you must define the number of scheduled payments and their respective amounts. There is no limit to the number of scheduled payments that may be set up under a pay plan.

Automatic Payments Can Be Created On The Scheduled Payment Dates

The system will create automatic payments on a pay plan's scheduled payment dates if:

- The account is set up for automatic payment (as described under [How To Set Up A Customer To Pay Automatically](#)), and
- The payment method defined on the pay plan indicates automatic payment is being used

The background process called PPAPAY is responsible for creating these automatic payments. It does this by calling the automatic payment creation algorithm plugged in on the installation record.

NOTE:

If the **Autopay Creation Option** on the [installation record](#) is set to Create On Extract Date, the automatic payment is NOT distributed and frozen when the automatic payment is initially created. Rather, a separate background process ([APAYDSFR](#)) distributes and freezes the automatic payment on the automatic payment GL distribution date (refer to [Automatic Payment Dates](#) for more information on how this date is calculated). Refer to [Automatic Payments](#) for more information.

A Pay Plan Insulates Overdue Debt From The Account Debt Monitor (ADM)

A pay plan's scheduled payments are used by the account debt monitor as "pseudo payments" that relieve the account's debt before it is subjected to the collection criteria (refer to [How Does The Account Debt Monitor Work](#) for more information about collection criteria).

It's important to understand that a pay plan only insulates the account's debt that belongs to the pay plan's debt class. Therefore, if a customer has debt that belongs to two debt classes (e.g., normal debt and 3rd party pass through debt), you would need to set up a separate pay plan for each debt class (assuming both types of debt are covered by a pay plan). Refer to [Different Collection Criteria For Different Customers and Different Debt](#) for more information about debt classes.

A Pay Plan Must Reference A Pay Plan Type

When you create a pay plan, you must define its pay plan type. The pay plan type controls the following functions:

- The debt class whose debt is insulated by the pay plan.
- The type of algorithm (if any) that is executed when the pay plan is broken. You might use such an algorithm to affect the customer's credit rating when the pay plan is broken.

A Pay Plan May Reference A Third-Party Payor

In addition to referencing the account whose debt is insulated by the pay plan, the pay plan must also reference the account that is responsible for making the payments. We refer to this second account as the pay plan's "payor".

While the payor's account is typically the same as the account whose debt is insulated by the pay plan, you can indicate a third-party payor (e.g., a social service agency) is responsible for making the pay plan's scheduled payments.

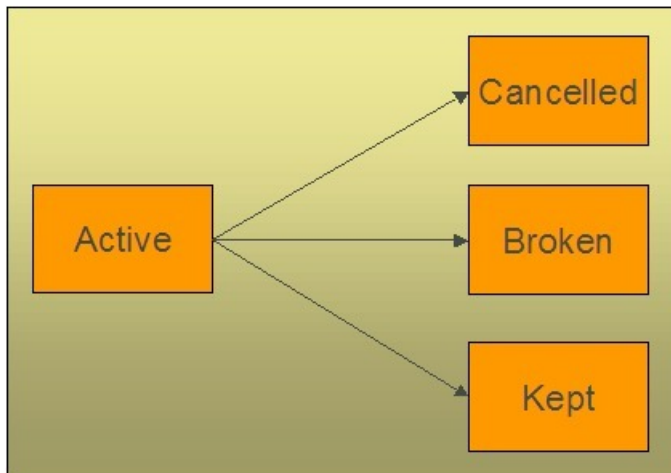
If your organization allows third-party payors, you can define each on the third-party payor control table. This control table exists to simplify the data-entry effort when you create a pay plan (as it defines the account associated with the third-party payor).

NOTE:

If a pay plan does not reference a third-party payor, any non-third-party payor (i.e., any account that is not defined in the third-party payor control table) can make payments on behalf of the customer. If a pay plan references a third-party payor, only payments made by the third party on behalf of the customer are counted towards the fulfillment of the pay plan.

The Lifecycle Of A Pay Plan

The following diagram shows the possible lifecycle of a pay plan:



The following points explain this lifecycle:

- Pay plans are initially created in the active state. Active pay plans are monitored for compliance by [The Pay Plan Monitor](#).

- A pay plan may be cancelled as follows:
 - A user can cancel a pay plan at will.
 - **When a SA is stopped AND there are no other active SAs in the same debt class, all active pay plans associated with the account and debt class will be cancelled.**
 - The activation of a collection event that calls the "cancel pay plan" algorithm will cancel all active pay plans associated with the collection process's debt class. You may want to use such a collection event if your organization cancels active pay plans when new debt causes a collection process to kick-off. Note, the base package algorithm that performs this function will not cancel the pay plan if it's associated with a 3rd party payor.
- [The Pay Plan Monitor](#) causes active pay plans to become broken if sufficient payments have not been made to satisfy the pay plan's scheduled payments.
- [The Pay Plan Monitor](#) causes active pay plans to become kept when it detects that sufficient payments have been made to satisfy the pay plan's scheduled payments.

Highlighting The Existence Of Broken / Kept / Active and Denied Pay Plans

You can define on the installation record plug-in algorithms that format alert messages. (Refer to [Installation Options - Algorithms](#) for additional information.) We recommend that you take advantage of the following algorithms to highlight pay plans:

- Highlight pay plans in a given status. This algorithm is used to highlight pay plans in a given state (broken, kept, cancelled) that were started within the last X days.
- Highlight customer contacts of a given type. This algorithm would be used to highlight customer contacts of a given type that were created within the last X days. This would be useful if you create a specific type of customer contact when you deny a pay plan. Some utilities do this to prevent customers from shopping around.

In addition, you can define account-specific alerts to highlight customers that should never be allowed to have a pay plan (for whatever reason).

A Pay Plan Must Reference A Payment Method

When you create a pay plan, you must define how the customer will make the payments be referencing a payment method. Examples of payment methods include: In Person , Wire Transfer, By Post, Express Mail, etc.

The payment method is more than just documentation as it defines the number of grace days the customer has to make the pay plan's scheduled payments. For example, if you set up the payment method control table to indicate that payments made By Post have 3 grace days, then the customer has up to 3 days after each scheduled payment date to make the payment. If payment is not received by the scheduled payment date plus the grace days, the pay plan will be marked as broken (and the ADM will be triggered).

The Pay Plan Monitor

FASTPATH:

Please understand the concepts described in [The Lifecycle Of A Pay Plan](#) and [The Tendering Account May Differ From The Account Whose Debt Is Relieved](#) before reading this section.

The Pay Plan Monitor background process (referred to as PPM) is responsible for monitoring active payment plans. This process can cause a pay plan (PP) to become kept or broken (or being left as active).

NOTE:

When is a pay plan marked as broken / kept? It's important to understand that only the PPM can cause a pay plan to become kept or broken. This means that if a customer makes a payment that satisfies a pay plan, the pay plan will only be marked as kept when the pay plan monitor next runs. Analogously, if a payment is cancelled, nothing will happen to an active pay plan until the PPM next runs. When the PPM next runs, it will see that the scheduled payment was not kept and it will break the pay plan and schedule the ADM to be executed. When the ADM next executes, it will create a collection process (because the customer's debt will no longer be insulated by the pay plan's scheduled payments).

NSF Cancellations After A Pay Plan Is Kept. If a payment is cancelled due to non-sufficient funds (NSF) after a pay plan is marked as kept, the pay plan will remain kept. But keep in mind that the pay plan's account is scheduled for review by the ADM when a payment is cancelled due to NSF. When the ADM reviews the account's debt, it will no longer have an active pay plan to insulate it and the account's debt will likely trigger a new collection process. Refer to [How Pay Plans Affect The ADM](#) for more information.

The following points describe, at a high level, how the PPM monitors a pay plan (PP) for compliance.

- The system selects all frozen, non-cancelled payment segments associated with the PP's account and debt class where:
 - The payment date is after the start date of the pay plan, and
 - The payment's pay event has at least one tender that references the pay plan's payor.
 - The system logically reduces / removes past and current scheduled payments (starting with the earliest scheduled payment) until the total amount of payment segments is exhausted (or there are no more historical / current scheduled payments).
-

NOTE:

Paying pay plans in advance. Scheduled payments with a future date are not logically removed / reduced. This means that if a customer makes advance payments on a pay plan, it will not be marked as kept until all scheduled payment dates are in the past.

- If all scheduled payments have been logically removed, the pay plan is marked as kept.
 - If there exist scheduled payments where the pay date + grace days (grace days are defined on the pay plan's payment method) is before the current date (i.e., a payment doesn't exist for a scheduled payment):
 - The pay plan is marked as broken.
 - The PP's break algorithm (if any) is called (note, for European / Australian pay plans, there are scenarios where the break algorithm can cause the pay plan to become unbroken - when there aren't at least two missed, historical scheduled payments).
 - An ADM trigger is stored for the PP's account. This will cause the account to be reviewed by the ADM the next time it runs. And because the pay plan is broken, its scheduled payments will no longer insulate the account's arrearage.
-

IMPORTANT:

It's important that you schedule the PPM to run before the ADM so that it can break unpaid payment plans prior to the ADM subjecting the account's debt to collection criteria. Refer to [How Pay Plans Affect The ADM](#) for more information.

How Pay Plans Affect The ADM

As described under [A Pay Plan Insulates Overdue Debt](#), a pay plan's scheduled payments insulate an account's debt from the ADM. This section describes how this is accomplished.

WARNING:

You should understand the concepts in [How Does The Account Debt Monitor Work](#) and [The Tendering Account May Differ From The Account Whose Debt Is Relieved](#) before reading the following.

NOTE:

The ADM will be triggered when a pay plan is broken. Refer to [The Pay Plan Monitor](#) for an explanation of how the ADM is triggered when a pay plan is broken.

Before the ADM (and ADM2) subjects an account's debt to the collection criteria, it calls the debt's debt class's Override Arrears Algorithm (this is an optional plug-in spot on [Debt Class](#)). This algorithm is passed the debt class's aged debt and manipulates it as follows:

- First, a list of all past, present and future scheduled payments associated with the account and debt class's active pay plans is constructed.
 - If multiple payors are encountered (because the customer has multiple pay plans and these have different payors), a separate list of scheduled payments is maintained for each payor.
- Next, for each payor, retrieve the total amount of frozen, non-cancelled payment segments made on behalf of the pay plan's account and debt class.
 - Select all frozen, non-cancelled payment segments associated with the pay plan's account and debt class whose pay date is \geq pay plan's start date and the pay segment's event has at least one tendering account associated with the pay plan's payor.
- Next, logically reduce / remove past and current scheduled payments (starting with the earliest scheduled payment) until the payor's payment amount is exhausted (or there are no more historical / current scheduled payments). Future scheduled payments cannot be remove / reduced.
- Finally, reduce the passed in aged debt with any unpaid scheduled payments.

NOTE:

This logic is not "hard coded". Rather, the mechanism used to use a pay plan's scheduled payments to reduce debt is defined in an algorithm defined on the pay plan's debt class. The contents in this section describe how a base package algorithm works. Because it's an algorithm, you can introduce whatever logic you please.

The following is an example of how pay plans affect aged debt.

Date	Event	SA's Arrears	SA's Balances	Scheduled Payments
Prior to creation of the PP 1/18/2000		\$1,000 - 90 days old	Current: \$4,500	
		\$1,600 - 60 days old	Payoff: \$4,500	
		\$1,900 - 30 days old		
1/18/2000	Pay Plan created.	\$1,000 - 90 days old	Current: \$4,500	1/20/2001 \$1,500
	The \$4,500 in future scheduled payments offsets the existing \$4,500 of aged debt.	\$1,600 - 60 days old	Payoff: \$4,500	2/01/2001 \$1,500
		\$1,900 - 30 days old		2/07/2001 \$1,500
		De facto ADM debt: \$0		
1/20/2001	The customer pays \$1,500. There exists \$3,000 of future scheduled payments that offset the arrears	\$1,100 - 62 days old	Current: \$3,000	1/20/2001 \$1,500 "Paid"
		\$1,900 - 32 days old	Payoff: \$3,000	2/01/2001 \$1,500 Future
		De facto ADM debt: \$0		2/07/2001 \$1,500 Future
1/20/2001	ADM runs. The \$3000 in future scheduled	\$1,100 - 62 days old	Current: \$3,000	1/20/2001 \$1,500 "Paid"
		\$1,900 - 32 days old	Payoff: \$3,000	2/01/2001 \$1,500 Future

	payments offsets the Current Balance of \$3000, so CC events not created.	De facto ADM debt: \$0		2/07/2001 \$1,500 Future
1/24/2001	A new bill is created for \$400	\$1,100 - 62 days old \$1,900 - 32 days old \$400 - 1 day old	Current: \$3,400 Payoff: \$3,400	1/20/2001 \$1,500 "Paid" 2/01/2001 \$1,500 Future 2/07/2001 \$1,500 Future
		De facto ADM debt: \$400 - 1 day old		
2/2/2001	Pay Plan Monitor runs. PP marked as Broken because the 2/1/2001 scheduled payment has not been paid (assuming no grace period on the pay plan's payment method)	\$1,100 - 74 days old \$1,900 - 44 days old \$400 - 8 days old	Current: \$3,400 Payoff: \$3,400	1/20/2001 \$1,500 "Paid" 2/01/2001 \$1,500 "Late" 2/07/2001 \$1,500 Future
2/2/2001	ADM runs. There are no active pay plans and therefore there is nothing to insulate the customer's debt. Therefore the aged debt will be subjected to the collection criteria and an appropriate collection process will be created.	\$1,100 - 74 days old \$1,900 - 44 days old \$400 - 8 days old	Current: \$3,400 Payoff: \$3,400	Pay plan is broken and therefore its scheduled payments cannot be used.
		De facto ADM debt is the same as above (i.e., rather old)		

Collection Process / Severance Process Cancellation

When a pay plan (PP) is created, the system determines if it can cancel active collection and severance processes associated with the pay plan's account and debt class. It does this because a pay plan's scheduled payments act as "pseudo payments" that relieve the account's debt (temporarily). The following points describe how this works:

- The system attempts to cancel collection processes by calling the [Collection Process Cancel Criteria Algorithm](#) defined on the debt class that is associated with the collection process. This algorithm is meant to cancel a collection process if the sum of ALL service agreements in the debt class have debt less than a given threshold amount. Because of the existence of the pay plan, the actual debt will be temporarily reduced by the amount of the pay plan's scheduled payments before it is compared to the threshold amount (see [How Pay Plans Affect The ADM](#) for more information about how debt is reduced). Note: this temporary reduction will only occur if you have plugged in the appropriate pay plan debt reduction algorithm on the debt class.
- It attempts to cancel severance processes by calling the [Severance Process Cancel Criteria Algorithm](#) defined on the debt class that is associated with the severance process. This algorithm is meant to cancel a severance process if the sum of ALL service agreements in the debt class have debt less than a given threshold amount. Because of the existence of the pay plan, the actual debt will be temporarily reduced by the amount of the pay plan's scheduled payments before it is compared to the threshold amount (see [How Pay Plans Affect The ADM](#) for more information about how debt is reduced). Note: this temporary reduction will only occur if you have plugged in the appropriate pay plan debt reduction algorithm on the debt class.

If collection / severance processes still exist for the account / debt class associated with the pay plan, a warning is issued.

Interesting Pay Plan Facts

The following points describe a variety of interesting facts about pay plans (PP):

- An account may have many active pay plans. However, only 1 pay plan may be active for a given account / debt class / payor at any point in time.
- The existence of a pay plan has no impact on payment distribution.
- When a SA is stopped, if the SA's debt class has an active PP AND there are no other active SAs in the same debt class, the PP will be cancelled
 - The cancel reason will be "cancelled by system" (as opposed to "cancelled by user")
- If necessary, different collection / severance processes can be triggered if a broken PP is detected (via the override algorithms on CCC and write-off control - we do NOT provide such algorithms).

Setting Up Pay Plan Control Tables

This section describes the control tables needed to set up pay plans.

Setting Up Pay Plan Types

Pay plan types control what is done by a given pay plan. Open **Admin > Credit & Collection > Pay Plan Type** to define your pay plan types.

FASTPATH:

For more information refer to [The Big Picture Of Pay Plans](#) for more information.

Description of Page

To modify a pay plan type, simply move to a field and change its value. To add a new pay plan type, press + to insert a row, then fill in the information for each field. The following fields display:

Pay Plan Type The name of the pay plan type.

Description A meaningful description of the pay plan type.

Broken Algorithm This algorithm is called when a pay plan is broken. Refer to [The Pay Plan Monitor](#) for more information about how pay plans are broken.

If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that breaks a pay plan. Click [here](#) to see the algorithm types available for this plug-in spot.

Debt Class The debt class covered by pay plans of this type. Refer to [Setting Up Debt Classes](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PP_TYPE](#).

Setting Up Payment Methods

Payment methods are used to describe how a customer intends to make their pay plan's scheduled payments. Open **Admin > Credit & Collection > Pay Method** to define your payment methods.

FASTPATH:

For more information refer to [A Pay Plan Must Reference A Payment Method](#) for more information.

Description of Page

To modify a pay method, simply move to a field and change its value. To add a new pay method, press + to insert a row, then fill in the information for each field. The following fields display:

Pay Method The name of the payment method.

Description A meaningful description of the payment method.

Grace Days The number of days added to the scheduled payment date. The ADM will consider the pay plan to be broken if payment is not made by the scheduled date plus the grace days.

Auto Pay If the pay method is marked as being for **Auto Pay**, the PPAPAY background process will automatically create an automatic payment on the pay plan's scheduled payment dates IF the account has been set up for automatic payment.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PAY_METH](#).

Setting Up Third Party Payors

Pay plans support optional third-party payors. Open **Admin > Credit & Collection > Third Party Payor** to define your third-party payors.

NOTE:

A third-party payor refers to an account. You must set up the account before you can create a third-party payor.

FASTPATH:

Refer to [A Pay Plan May Reference A 3rd Party Payor](#) for more information.

Description of Page

The following fields display for each third party payor:

Third Party Payor Provide a meaning id for the third-party payor that can be easily recognized when setting up a pay plan.

Description A meaningful description of the payor.

Account ID The account that is used for this payor. It is this account that is tracked as the "payee" of any payments made towards a third party payor's pay plan.

Active Check this box if the payor is currently available to participate in pay plans.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_THRD_PTY](#).

Creating Collection, Severance & Write-Off Procedures

Your collection procedures define how your organization collects overdue debt. Your severance procedures define how your organization severs service agreements when collection attempts fail. Your write-off procedures define how your organization writes off finalized debt. In this section, we describe how to set up the data that controls these procedures.

FASTPATH:

For more information about collection, severance and write-off procedures, see [The Big Picture Of Credit & Collections \(C&C\)](#).

WARNING:

There are innumerable ways to design your collection, severance and write-off procedures. Some designs will result in easy long-term maintenance, others will result in maintenance headaches. In this section, we provide information to help you understand the ramifications of the various options. Before you set up your production collection, severance and write-off procedures, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

Designing Your Collection Procedures

The design of your collection procedures is an iterative process. Over time, you will develop intuitive skills that will allow you to skip some iterations. However, when you're starting out, we recommend you use the following matrix as your guide. When the matrix is complete, you're ready to set up the collection process control tables.

SA's Debt Class	Account's Collection Class:	Account's Collection Class:

The topics discussed below will gradually complete this matrix using a simple case study.

FASTPATH:

For more information about how the information in this matrix is used to monitor your customers' debt, refer to [Different Collection Criteria For Different Customers And Different Debt](#).

Designing Your Debt Classes

Multiple debt classes are needed when you have different collection procedures for different types of service agreements. If all service agreement debt is collected the same way, then you'll have just one debt class (call it Generic). However, if you're like many organizations, you will have multiple debt classes. The following points will help you understand why:

- If you have both regulated and unregulated service, you probably have different debt classes for each type of service. Why? Because your local regulators control how you collect and cutoff overdue, regulated debt. For unregulated debt, your organization controls how overdue debt is collected.
- If your customers make charitable donations, you will have a charitable contribution debt class. Why? Because you probably send a different type of letter when the customer falls into arrears on their charitable contributions. You also can't cut off their water service if they don't make their charitable contributions.
- If you levy deposits, you will probably have a deposit debt class. Why? Because you probably respond differently if the customer doesn't pay their deposit (e.g., you may decide to cut off their electric service until the deposit is paid).
- If you allow customers to make payments on non-billed budgets, you will probably have a non-billed budget debt class. Why? Because you probably respond differently if the customer doesn't pay their non-billed budget (e.g., you may decide to expire the non-billed budget but not affect their other service since the non-billed budget is a way for customers to prepay for upcoming bills).

- If you write-off uncollectable debt, you will need another debt class for write-off service agreements. Why? Because when you write-off debt in the system, you transfer the uncollectable debt from the original service agreement(s) to a write-off service agreement. The write-off service agreement holds this debt forever (or until it is paid). You need to use a different debt class for the write-off service agreements because they aren't subject to collection criteria.
- If you use the system to charge your organization's company usage, you'll need another debt class (we refer to it as the "N/A" debt class below). Why? Because all service agreements must have a debt class, even those that will never have debt.

SA's Debt Class	Account's Collection Class	Account's Collection Class
Regulated		
Unregulated		
Charitable Contribution		
Deposit		
Non-Billed Budget		
Write Off		
N/A		

Designing Your Collection Classes

Multiple collection classes are needed when any debt class has different collection rules depending on the type of customer. If all customers within all debt classes are collected the same way, then you'll just have a single collection class (call it Generic). However, if you're like many organizations, you will have multiple collection classes.

Consider unregulated debt. For commercial/industrial customers, you probably don't worry until they owe you more than, say, \$100 after 20 days. For residential customers, you probably don't worry until they owe you more than, say, \$5 after 20 days. In this situation, you will have at least two collection classes: one for large customers, the other for residential customers.

In our example, we are assuming you have two collection classes: Residential and Commercial/Industrial.

SA's Debt Class	Account's Collection Class:	Account's Collection Class:
	Residential	Commercial/Industrial
Charitable Contribution		
Regulated		
Unregulated		
Write Off		
Company Usage		

Designing Collection Class Controls

At this point we have the rows and columns defined in our matrix. Now it's time to work on the individual cells.

Each cell should have a "collection class control" that defines its collection criteria and what to do if the criteria are violated. If a cell doesn't have a collection class control, this means you don't have any debt associated with that combination of collection class and debt class. So, we'll mark each cell without debt with "N/A".

SA's Debt Class	Account's Collection Class:	
	Residential	Commercial/Industrial
Regulated		N/A
Unregulated		
Charitable Contribution		N/A
Deposit		
Write Off		
N/A		

Next, we'll mark each cell for debt classes whose debt isn't collectable (i.e., the write-off and N/A debt classes).

SA's Debt Class	Account's Collection Class:	
	Residential	Commercial/Industrial
Regulated		N/A
Unregulated		
Charitable Contribution		N/A
Deposit		
Write Off	N/A	N/A
N/A	N/A	N/A

NOTE:

If the Account Debt Monitor encounters debt associated with a non-existent collection class control, it will issue an error.

Determining the collection criteria in each remaining cell can be straightforward or complicated; it depends on how your organization works. Our case study assumes the following:

- For charitable debt, if the customer is more than \$0 in arrears by more than 20 days, kick off the "charity reminder" collection process. We'll talk more about this collection process later.
- For regulated / residential debt, if the customer is more than \$15 in arrears by more than 20 days, kick off the Normal Regulated collection process. We'll talk more about this collection process later.
- For unregulated / residential debt, if the customer is more than \$5 in arrears by more than 20 days, kick off the Normal Unregulated collection process. We'll talk more about this collection process later.
- For unregulated / commercial-industrial debt we have multiple criteria:
 - Highest priority. If the customer is more than \$10,000 in arrears by more than 20 days, kick off the Large Overdue Debt collection process. We'll talk more about this collection process later.
 - Lower priority. If the customer is more than \$100 in arrears by more than 20 days, kick off the Normal Unregulated collection process. We'll talk more about this collection process later.
- For deposit debt (regardless of collection class) we have multiple criteria:
 - Highest priority. If the customer is more than \$5 in arrears by more than 50 days, kick off the Deposit Severely Overdue collection process. We'll talk more about this collection process later.
 - Lower priority. If the customer is more than \$15 in arrears by more than 20 days, kick off the Deposit collection process. We'll talk more about this collection process later.

Given the above, our matrix will look as follows:

SA's Debt Class	Account's Collection Class:	Account's Collection Class:
	Residential	Commercial/Industrial
Charitable Contribution	If > \$0 is older than 20 days, start Charity Reminder collection process.	N/A
Regulated	If > \$15 is older than 20 days, start Normal Regulated collection process.	N/A
Unregulated	If > \$5 is older than 20 days, start Normal Unregulated collection process.	Highest priority: If > \$10,000 is older than 20 days, start Large Overdue Debt collection process. Lower priority: If > \$100 is older than 20 days, start Normal Unregulated collection process.
Deposit	Highest priority: If > \$5 is older than 50 days, start Deposit Severely Overdue collection process. Lower priority: If > \$15 is older than 20 days, start Deposit collection process.	Highest priority: If > \$5 is older than 50 days, start Deposit Severely Overdue collection process. Lower priority: If > \$15 is older than 20 days, start Deposit collection process.
Write Off	N/A	N/A
N/A	N/A	N/A

Designing Your Collection Class Control Overrides

WARNING:

Your collection needs may not require any overrides for your collection class control matrix and therefore this section may not be relevant.

The following matrix will help you design your collection class overrides. When the matrix is complete, you're ready to set up the collection class control tables.

Notice that the matrix has two dimensions: one is dependent on collection condition algorithms; the other is dependent on the collection class controls designed in the previous section. Collection condition algorithms are confusing. Think of them as optional conditions that, if met, will subject the collection class control's debt to different collection criteria.

Each cell in the matrix contains the collection criteria that will be applied to the account's debt when the collection condition is met (i.e., the same type of criteria - dollars and days and collection process - are defined in each cell).

We label the first collection condition as the Default. The collection criteria associated with this column will be used to analyze an account's debt when none of the other conditions applies. We'll start by indicating the Default collection criteria (this was defined in the previous section).

SA's Debt Class	Account's Collection Class:	Account's Collection Class:
	Default	Credit Rating < Threshold
Residential Charitable Contribution	See default collection criteria defined in previous section.	
Residential Regulated	See default collection criteria defined in previous section.	
Residential Unregulated	See default collection criteria defined in previous section.	

Commercial-Industrial Unregulated	See default collection criteria defined in previous section.
Residential Deposit	See default collection criteria defined in previous section.
Commercial-Industrial Deposit	See default collection criteria defined in previous section.

If a different collection process OR criteria should be used when other conditions are met, you should indicate such by defining the collection criteria in the cell. For example, if we assume that all unregulated residential debt has a different collection process when the account's credit score is less than the threshold credit rating on the installation record, our matrix will look as follows:

SA's Debt Class	Account's Collection Class:	
	Default	Credit Rating < Threshold
Residential Charitable Contribution	See default collection criteria defined in previous section.	
Residential Regulated	See default collection criteria defined in previous section.	
Residential Unregulated	See default collection criteria defined in previous section.	Override: If Credit Rating is lower than the installation threshold: If > \$5 is older than 15 days, start Risky Unregulated collection process.
Commercial-Industrial Unregulated	See default collection criteria defined in previous section.	
Residential Deposit	See default collection criteria defined in previous section.	
Commercial-Industrial Deposit	See default collection criteria defined in previous section.	

Once the matrix is complete, you're ready to design your collection process and collection event types.

NOTE:

The collection conditions are limited by your imagination (and business requirements). We have provided the collection conditions you see above as an example; we don't expect you'll be able to use the exact conditions we supply. Your conditions will be based on any number of factors. For example, if you have different collection criteria that apply during winter months, you should add a new collection condition (called Winter Season). Or if you have different criteria based on years of service, you could have another condition.

New collection conditions may require programming. See [How To Add A New Algorithm](#) for more information.

Designing Collection Process Templates & Collection Event Types

The following table shows the collection process templates referenced in the previous section's matrix. Adjacent to each process are its events and an indication of when they are triggered.

Collection Process Template	Collection Event Template	Triggered X Days From Start Of Collection Process
Charitable Contribution Reminder	Charity courtesy reminder letter	0

	Start severance	15
Normal Regulated	Regulated courtesy reminder letter	0
	Regulated second notice letter	10
	Start severance	15
Large Overdue Debt	Large debt courtesy reminder letter	0
	To Do for large overdue debt	3
	Large debt second notice letter	10
	Start severance	15
Normal Unregulated	Unregulated courtesy reminder letter	0
	Unregulated second notice letter	5
	Start severance	10
Risky Customer Unregulated	Unregulated risky letter	0
	Start severance	5
Deposit	Deposit reminder	0
Deposit Severely Overdue	Create To Do entry	0

If we extract each unique event type from the above table, we end up with the following:

Collection Event Type	Event Type
Charity courtesy reminder letter	Send Letter - CHARIT REMIN
Start severance	Start Severance Process
Regulated courtesy reminder letter	Send Letter - REGUL REMIN
Regulated second notice letter	Send Letter - REGUL second
Large debt courtesy reminder	Send Letter - LARGE REMIN
Risky debt courtesy reminder	Send Letter - RISKY REMIN
To Do for large overdue debt	Issue To Do
Large debt second notice letter	Send Letter - LARGE second
Unregulated courtesy reminder letter	Send Letter - UNREG REMIN
Unregulated second notice letter	Send Letter - UNREG second
Deposit reminder	Send Letter - DEPOS REMIN
To Do for deposit severely overdue	Issue To Do

Now you're (almost) ready to set up your collection procedures.

Defining Cancellation Process Auto Cancellation Criteria

The topics in the section [How Are Collection Processes Cancelled](#) describe the two algorithms that play a part in the cancellation of a collection process. It also describes when to use what type of algorithm. Please read this section and then set up the appropriate cancellation criteria on your [Debt Classes](#), and optionally, on your [Collection Process Templates](#).

Setting Up Collection Procedures

In the previous section, [Designing Your Collection Procedures](#), we presented a case study that illustrated a mythical organization's collection procedures. In this section, we'll explain how to set up the control tables to implement these procedures:

Setting Up Collection Event Types

Collection event types control what is done by a given collection event. Open **Admin > Collection Event Type > Add** to define your collection event types.

Description of Page

Enter a unique **Collection Event Type** and **Description** for the collection event type.

Enter the **Collection Event Type**. Permissible values are: Affect Credit, Rating/Cash-Only, Cancel Budget, Generic Algorithm, Send Letter, Create To Do Entry, Start Severance Process. The following discussion describes the parameters that must be defined for each type of collection event.

The Affect Credit Rating/Cash-Only collection event type causes a credit rating demerit record to be linked to the account. This record is constructed using the following **Parameters**:

- Use **Credit Rating Points** to define this event's affect on the account's credit rating. This should be a negative number. An account's credit rating is equal to the start credit rating amount defined on the installation record plus the sum of credit rating demerits that are currently in effect. When an account's credit rating is less than the credit rating threshold defined on the installation record, the account's credit rating is displayed as an alert on Control Central.
- Use **Cash-Only Points** to define this event's affect on the account's cash-only score. This should be a positive number. When an account's cash-only score exceeds the cash-only threshold score defined on the installation record, the account is flagged as cash-only during payment processing and on Control Central.
- Use **Credit Rating Months** to define the length of time the demerit remains in effect. This information is used to define the effective period of the credit rating demerit record.

FASTPATH:

For more information, refer to [Account - Credit Rating](#).

The Send Letter collection event type causes a customer contact to be generated that, in turn, generates a letter. Enter the following **Parameters** for this type of event:

- Select the **Contact Class** used to categorize the customer contact.
- Use **Contact Type** to define the type of customer contact to create. The type of customer contact controls the type of letter that is generated.

NOTE:

Letter creation is triggered via a customer contact. You must set up a customer contact type for each type of letter you generate. You specify the necessary customer contact type on the collection event. Refer to [Setting Up Letter Templates](#) for more information.

The Cancel Budget collection event type cancel an account's budget plan (if the account is on such a plan). When a budget plan is cancelled, adjustments are issued to synchronize every service agreement's current balance with its payoff balance and each applicable SA's recurring charge amount (i.e., budget amount) is set to zero.

The Generic Algorithm collection event type causes the algorithm defined in the **Collection Event Alg** to be executed. You use this type of algorithm when the standard types of collection events won't do what you need done. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines the algorithm that will be called when events of this type of activated. Click [here](#) to see the algorithm types available for this plug-in spot.

The Create To Do Entry collection event type causes a To Do entry to be issued. A good example of where this is used is when the collection event requires that the customer be called on the phone. Refer to [The Big Picture of To Do Entries](#) for more information about To Do entries (refer to the To Do type TD-CEVT for the type of To Do entry that's created).

The Start Severance Process type will start a severance process for every service agreement linked to the collection process. No parameters are needed for this type of event.

Enter a **Long Description** to fully describe the collection event type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COLL_EVT_TYP](#).

Setting Up Collection Process Templates

Collection process templates define the collection events that will be executed when a collection criteria rule is violated. Open **Admin > Credit & Collection > Collection Process Template > Add** to define your collection process templates.

Description of Page

Enter a unique **Collection Process Template** and **Description** for the collection process template.

Select a **Cancel Criteria Algorithm** if your organization allows individual service agreements to be "removed" from a collection process regardless of the debt associated with all service agreements in the debt class. In other words, if your cancel criteria are based on the debt associated with ALL service agreements in a debt class DO NOT SPECIFY THIS ALGORITHM. If this algorithm is specified, it is executed by the collection process monitor when it detects that a credit has been applied to a service agreement linked to an active collection process. This algorithm will indicate if the specific service agreement that has been credited no longer has debt that warrants a collection process. Refer to [How Are Collection Processes Cancelled](#) for more information. If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that "removes" a service agreement from a collection processes if the service agreement's debt so warrants. Click [here](#) to see the algorithm types available for this system event.

The **Response** grid contains an entry for every collection event that will be created when a collection process that references this template is created. The following information must be defined for each event:

Event Sequence controls the order in which the collection event types appear under the collection process template. The sequence number is system-assigned and cannot be changed. If you have to insert a collection event type between two existing templates, you'll have to remove the latter events, insert the new event, and then re-specify the removed events.

Collection Event Type Specify the type of collection event to be generated.

Days After Process Creation Specify the number of days after the creation of the collection process that the related collection event will be triggered. Refer to [Calendar vs Work Days](#) for a description of how this system uses this information to set the trigger date on the respective collection events.

FASTPATH:

For more information about collection event types, see [Setting Up Collection Event Types](#). For more information about trigger dates, see [Collection Event Trigger Date](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COLL_PROC_TM](#).

Setting Up Collection Classes

Every account has a collection class. This class is one of several fields that control the collection method applied to the account's debt. Open **Admin > Credit & Collection > Collection Class** to define your collection classes.

FASTPATH:

For more information about collection classes, see [Designing Your Collection Classes](#).

Description of Page

Enter a unique **Collection Class** code and **Description** for each collection class.

Indicate which method is used to monitor the member accounts' unpaid debt:

- If you practice [balance-forward accounting](#) for accounts belonging to this collection class, select Collection, Severance & Write-Off. This method of collection is described throughout this chapter.
- If you practice [open-item accounting](#) for accounts belonging to this collection class, select Overdue. This method of collection is described under [Defining Overdue Processing Options](#).
- If accounts belonging to this collection class are not subject to either of the above collection methods, select Not Eligible For Collection. Please be aware that these accounts will NOT be reviewed for overdue debt.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COLL_CL](#).

Setting Up Debt Classes

Every SA type has a debt class. This class is one of several fields that control the collection criteria applied to the service agreement's debt. Open **Admin > Credit & Collection > Debt Class > Add** to define your debt classes.

FASTPATH:

For more information about debt classes, see [Designing Your Debt Classes](#).

Description of Page

Enter a unique **Debt Class** and **Description** for the debt class.

Turn on **Eligible for Collection** if service agreements belonging to this debt class have their debt monitored by the collection process. This should only be turned off if this debt cannot be collected, e.g., write-off debt.

The grid that follows contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to an arbitrary number as only one algorithm for each system event is allowed in this case.

WARNING:

These algorithms are typically significant system processes. The absence of an algorithm may prevent the system from operating correctly.

You can define algorithms for the following **System Events**:

System Event	Optional / Required	Description
Collection Process Cancellation Rule	Required if debt class is eligible for collection	<p>This algorithm determines if a collection process can be canceled, and if so, it cancels it. Refer to How Are Collection Processes Cancelled for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Severance Process Cancellation Rule	Required if debt class is eligible for collection	<p>This algorithm determines if a severance process can be canceled, and if so, it cancels it. Refer to How Are Severance Processes Cancelled for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Override Arrears Due To Pay Plan	Required if you use Pay Plans	<p>This algorithm is called to temporarily override a customer's arrears using a pay plan's scheduled payments when the system looks at an account's debt from a credit & collections perspective (i.e., the ADM calls this algorithm before it subjects a customer's debt to the collection criteria and when a pay plan is created). It does not actually change any data, but overlays the current arrears with the pay plan scheduled payments.</p> <p>This algorithm is also called by the above algorithms when a pay plan is created in order to evaluate if the scheduled payments actually cover the arrears (if so, the collection / severance processes are cancelled). It is also called periodically by the ADM in order to establish if the current state of the pay plan still covers the arrears. Refer to How Pay Plans Affect The ADM.</p> <p>Click here to see the algorithm types available for this system event.</p>

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_DEBT_CL](#).

Setting Up Collection Class Controls

The topics in this section describe the windows on which you set up your collection class control information.

WARNING:

The information in this page is what controls how the system analyzes your customer's debt. The flexibility of this control information provides you with almost unlimited options. This is very powerful, but it requires careful analysis. Refer to [Designing Your Collection Procedures](#) for more information.

Collection Class Control - Main Information

The information on this transaction defines the conditions that will be checked by the [Account Debt Monitor](#) when it checks if an account has violated your debt criteria.

Open **Admin > Credit & Collection > Collection Class Control > Add** to define this information.

FASTPATH:

For more information about collection class control, refer to [Designing Collection Class Controls](#).

Description of Page

Enter a unique **Collection Class Control** code and **Description** for the collection class control (CCC).

Enter the **CIS Division** to which the CCC's criteria applies.

Enter the **Collection Class** to which the CCC's criteria applies.

Enter the **Debt Class** to which the CCC's criteria applies.

Enter the **Currency Code** in which the CCC's criteria are denominated.

Use **Long Description** to further describe the CCC.

FASTPATH:

The information in the following grid is not intuitively obvious. Refer to [Designing Collection Class Controls](#) and [Designing Your Collection Class Control Overrides](#) for more information.

The grid which follows contains the conditions that are checked by the [Account Debt Monitor](#) (ADM) to determine the type of criteria (defined on the next tab) that will be applied against an account's debt. In other words - the ADM will check each condition (from highest to lowest **Priority**). The first condition that returns a value of true will cause the system to compare the account's debt against the debt criteria defined on the next tab.

Multiple conditions may be defined if different conditions result in a different type of debt thresholds (or a different type of collection process). The following fields are required for each condition:

Collection Condition Priority The priority controls the order in which the ADM checks if a collection condition applies (the lower the number, the higher the priority). Higher priorities are checked before lower priorities.

NOTE:

The values for this field are customizable using the Lookup table. This field name is COLL_CAT_PRIO_FLG.

Condition Algorithm Define the algorithm used to check if an account should be subject to the collection criteria defined on the next tab. If the algorithm returns a value of true (i.e., the condition is met), the ADM will compare the account's debt against the **Debt Criteria** (defined on the next tab) and start a collection process if the account has debt that violates these criteria.

You must have at least one collection condition; otherwise the system will not have criteria against which to compare the account's debt. This entry should have the lowest priority code and reference the "default" algorithm. If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that references the "default" collection condition algorithm type (**COLL COND DF**).

If you have other conditions that should be checked before the default condition, you must create an entry for each in this grid. Each entry should have a priority consistent with your business requirements (and this priority should be higher than the default condition's priority). In addition, you should reference an algorithm that contains the conditions that will be checked to determine if the account should be subject to the debt criteria (defined on the next tab). The system is supplied

with many additional algorithm types. In order to take advantage of them, you will need to create an algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that references one of the collection condition algorithm types. Click [here](#) to see the algorithm types available for this system event.

Where Used

Collection class controls contain the data that controls the Account Debt Monitor. Refer to [How Does The Account Debt Monitor Work?](#) for more information.

Collection Class Control - Debt Criteria

The information on this page defines the debt and age thresholds used by the [Account Debt Monitor](#) when it checks if an account has violated your acceptable levels of debt. Open **Admin > Credit & Collection > Collection Class Control > Search** and navigate to the **Debt Criteria** page to define this information.

NOTE:

The information on this page is not intuitively obvious. Refer to [Designing Collection Class Controls](#) and [Designing Your Collection Class Control Overrides](#) for more information.

Description of Page

The **Debt Criteria** scroll contains an entry for each collection criteria algorithm defined on the **Main** tab. The following information appears

The **Collection Condition Priority** controls the order in which the Account Debt Monitor (ADM) checks if a collection condition applies. Higher priorities are checked before lower priorities.

The **Condition Algorithm** is called by the ADM to determine which collection criteria should be applied to the account's debt. If this algorithm returns a value of true (i.e., the condition is met), the ADM will compare the account's debt against the **Debt Criteria** defined below. If the account violates any criteria, a collection process will be started (using the respective **Collection Process Template**).

The grid that follows contains the debt age and amount of debt that must be violated by the account in order for the ADM to create a collection process template. The following fields should be defined:

Arrears Priority controls the order in which the arrears criteria will be checked by the Account Debt Monitor (the lower the number, the higher the priority). The first criteria, if any, that is met will cause a collection process to be created (using the **Collection Process Template**).

NOTE:

The values for this field are customizable using the Lookup table. This field name is CRIT_PRIO_FLG. Be aware that this field is used for multiple tables: [Collection Class Control](#), [Severance Criteria](#), [Write Off Control](#) and [Workflow Process Profiles](#).

Collection Process Template If the Account Debt Monitor determines that the account's debt violates the corresponding criteria, it creates a collection process using the specified collection process template.

Arrears Amount When the Account Debt Monitor checks an account's debt, it determines if the account has debt older than "> Number of Days" (the next field) AND the debt exceeds "> Arrears Amount". If so, a collection process is started.

Days When the Account Debt Monitor checks an account's debt, it determines if the account has debt older than **Days** AND the debt exceeds **Arrears Amount**. If so, a collection process is started.

Where Used

Collection class controls contains the data that controls the [Account Debt Monitor](#).

Designing Your Severance Procedures

The following matrix will help you design your severance procedures. When the matrix is complete, you're ready to set up the severance process control tables.

Notice that the matrix has two dimensions: one is dependent on severance criteria algorithms; the other is dependent on the SA type of the service agreement being severed. The number and type of SA types is dependent on how your organization sets up the SA type table (the SA types shown below are characteristic of those used by a simple utility).

SA Type	Severance Criteria Algorithm:	Severance Criteria Algorithm:
	Default	Customer On Life Support
Electric Residential		
Electric Commercial		
Gas Residential		
Gas Commercial		
Charitable Contribution		

Once you know the values of each dimension, you fill in each cell with its respective severance events. We've completed the sample matrix with some characteristic events.

SA Type	Severance Criteria Algorithm:	Severance Criteria Algorithm:
	Default	Customer On Life Support
Electric Residential	<p>Create a 48-hour warning field activity.</p> <p>2 days after completion, create a disconnect service field activity.</p> <p>Immediately after completion of the disconnect field activity, send a letter to the customer.</p> <p>10 days after completion of disconnection, expire the service agreement.</p>	<p>Create a To Do entry asking a collection rep to call the customer.</p> <p>5 days later, create a 72-hour warning field activity.</p> <p>2 days after completion, create a To Do entry telling collection rep of impending life support cutoff.</p> <p>3 days after completion of warning, create a disconnect service field activity AND generate a To Do entry informing a collection agent of such.</p> <p>Immediately after completion of the disconnect field activity, send a letter to the customer.</p> <p>10 days after completion of disconnection expire the service agreement.</p>
Electric Commercial	<p>Create a 48-hour warning field activity.</p> <p>2 days after completion, create a disconnect service field activity.</p> <p>Immediately after completion of the disconnect field activity, send a letter to the customer.</p> <p>10 days after completion of disconnection, expire the service agreement.</p>	N/A

Gas Residential	<p>Create a 48-hour warning field activity.</p> <p>2 days after completion, create a disconnect service field activity.</p> <p>Immediately after completion of the disconnect field activity, send a letter to the customer.</p> <p>10 days after completion of disconnection, expire the service agreement.</p>	<p>Create a 48-hour warning field activity.</p> <p>2 days after completion, create a disconnect service field activity.</p> <p>Immediately after completion of the disconnect field activity, send a letter to the customer.</p> <p>10 days after completion of disconnection, expire the service agreement.</p>
Gas Commercial	<p>Create a 48-hour warning field activity.</p> <p>2 days after completion, create a disconnect service field activity.</p> <p>Immediately after completion of the disconnect field activity, send a letter to the customer.</p> <p>10 days after completion of disconnection, expire the service agreement.</p>	N/A
Charitable Contribution	Expire service agreement	Expire service agreement

Once the matrix is complete, you determine the severance process templates needed to implement your severance procedures. The following table shows the severance process templates referenced in the previous section's matrix. Adjacent to each process are its events and an indication of when they are triggered.

Severance Process Template	Event Number	Severance Event Template	Dependent On Event(s)	Trigger Date Set To X Days After Completion Of Dependent Events
Utility severance - default	10	Field activity - 48 hour disconnect for non-payment warning	N/A - first event	0
	20	Field activity - disconnect for non-payment	10	2
	30	'Service has been disconnected' letter	20	0
	40	Expire service agreement	20	10
Electric life support residential severance	10	Generate delinquent life support customer To Do entry	N/A - first event	0
	20	Field activity - 72 hour disconnect for non-payment warning	10	5
	30	Generate impending life support cutoff To Do entry to C&C rep	20	2
	40	Field activity - cut for non-payment	20	3
	50	Service has been disconnected letter	40	0
	60	Expire service agreement	40	10

If we extract each unique severance event type from the above table, we end up with the following:

Severance Event Template	Event Type
48-hour warning	Generate Field Activity - Disconnect Warning
72-hour warning	Generate Field Activity - Disconnect Warning
Disconnect for non payment	Generate Field Activity - Cut For Non-Payment
Delinquent life support customer	Create To Do Entry - C&C Rep Role
Impending life support cutoff	Create To Do Entry - C&C Rep Role
Service has been disconnected letter	Send Letter - Customer Contact Type is Disconnect Letter
Expire service agreement	Expire Service Agreement

WARNING:

The field activity types are NOT specified directly on the severance event type. Why? Because each service point linked to the service agreement being severed could necessitate a different type of field activity. Therefore, the system uses the type of service point, its state (e.g., connected, meter is off.) and the desired customer event (e.g., Disconnect Warning, Cut For Non-Payment) to determine which field activity type(s) to generate. Refer to [Setting Up Field Activity Type Profiles](#) for how to set up the specific disconnect field activity types for your various types of service points.

Now you're (almost) ready to set up your severance procedures.

Defining Severance Process Auto Cancellation Criteria

The topics in the section [How Are Severance Processes Cancelled](#) describe the two algorithms that play a part in the cancellation of a collection process. It also describes when to use what type of algorithm. Please read this section and then set up the appropriate cancellation criteria on your [Debt Classes](#), and optionally, on your [Severance Process Templates](#).

Designing Your Reconnection Procedures

If a customer pays for a service agreement after the service has been cut for non-payment AND BEFORE THE SA HAS BEEN EXPIRED, they need to be reconnected. Counter-intuitively, you must set up a severance process to initiate the field activities to reconnect service.

NOTE:

Why do you use a severance process to reconnect service? Because a severance process is nothing more than a series of events that take place one after another. Some of the events create field activities, others send letters, others create To Do entries. So, why not use a severance process? You just have to send different letters and perform different field activities.

WARNING:

The system will automatically create a reconnection process if a severance process is cancelled as a result of a payment (or other credits). Please note that this will only happen if you plug-in the appropriate post cancellation algorithm

on your severance process templates. Refer to [What Happens When A Severance Process Is Cancelled?](#) for more information.

While you don't define the reconnect procedures for an SA type, we recommend you think about the reconnection steps for each of your SA types that can be disconnected for nonpayment. We've completed the sample matrix with some characteristic events.

SA Type	Steps
Electric Residential	Create a reconnect service field activity. Immediately after completion of the reconnect, send a letter to the customer.
Electric Commercial	Create a reconnect service field activity. Immediately after completion of the reconnect, send a letter to the customer.
Gas Residential	Create a reconnect service field activity. Immediately after completion of the reconnect, send a letter to the customer.
Gas Commercial	Create a reconnect service field activity. Immediately after completion of the reconnect, send a letter to the customer.

Once the matrix is complete, you determine the severance process templates needed to implement your reconnection procedures. Notice each SA type has the same reconnection steps. This means you just need one severance process. The following table shows this severance process template and its events.

Severance Process Template	Event Number	Severance Event Template	Dependent On Event(s)	Trigger Date Set To X Days After Completion Of Dependent Events
Reconnect	10	Field activity - reconnect service	N/A - first event	0
	20	Service has been reconnected letter	10	0

If we extract each unique severance event type from the above table, we end up with the following:

Severance Event Template	Event Type
Reconnect	Generate Field Activity - Reconnect
Service has been reconnected letter	Send Letter

WARNING:

The field activity types are NOT specified directly on the severance event type. Why? Because each service point linked to the service agreement being severed could necessitate a different type of field activity. Therefore, the system uses the type of service point, its state (e.g., connected, meter is off.) and the desired customer event (e.g., Disconnect Warning, Cut For Non-Payment) to determine which field activity type(s) to generate. Refer to [Setting Up Field Activity Type Profiles](#) for how to set up the specific disconnect field activity types for your various types of service points.

IMPORTANT:

If you want the system to automatically create a reconnection process if a customer pays after they have been cut, you must specify the appropriate post cancellation algorithm on your severance process templates.

And now you're ready to set up your severance (and reconnection) procedures.

Setting Up Severance Procedures

In the previous section, [Designing Your Severance Procedures](#), we presented a case study that illustrated a mythical organization's severance procedures. In this section, we'll explain how to set up the control tables to implement these procedures:

Setting Up Severance Event Types

Severance event types control what is done by a given severance event. Open **Admin > Credit & Collection > Severance Event Type > Add** to define your severance event types.

FASTPATH:

For more information refer to [Designing Your Severance Procedures](#).

Description of Page

Enter a unique **Severance Event Type** code and **Description** for the severance event type.

Enter the Severance **Event Type**. Permissible values are: Affect Credit Rating/Cash-Only, Send Letter, Generic Algorithm, Create To Do Entry, Create Field Activities, Expire Service Agreement. The following discussion describes the parameters that must be defined for each type of severance event.

The Affect Credit Rating/Cash-Only collection event type causes a credit rating demerit record to be linked to the account. This record is constructed using the following **Parameters**:

- Use **Credit Rating Points** to define this event's affect on the account's credit rating. This should be a negative number. A customer's credit rating is equal to the start credit rating amount defined on the installation record plus the sum of credit rating demerits that are currently in effect. When an account's credit rating is less than the credit rating threshold defined on the installation record, the account's credit rating is displayed as an alert on Control Central.
- Use **Cash-Only Points** to define this event's affect on the account's cash-only score. This should be a positive number. When an account's cash-only score exceeds the cash-only threshold score defined on the installation record, the account is flagged as cash-only during payment processing and on Control Central.
- Use **Credit Rating Months** to define the length of time the demerit remains in effect. This information is used to define the effective period of the credit rating demerit record.

The Send Letter severance event type causes a customer contact to be generated that, in turn, generates a letter. Enter the following **Parameters** for this type of event:

- Select the **Contact Class** used to categorize the customer contact.
- Use **Contact Type** to define the type of customer contact to create. The type of customer contact controls the type of letter that is generated.

The Generic Algorithm severance event type causes the algorithm defined in the **Sev. Event Algorithm** to be executed. You use this type of algorithm when the standard types of severance events won't do what you need done.

The Create To Do Entry severance event type causes a To Do entry to be created. Refer to [The Big Picture of To Do Entries](#) for more information about To Do entries (refer to the To Do type TD-SEVT for the type of To Do entry that's created).

The Generate Field Activities severance event type causes one or more field activities to be generated. Enter the following **Parameters** for this type of event:

- Select the **Customer Event** associated with the field activity. Valid values are: Cut For Non-Payment (CNP), Disconnect Warning (Disc Warn), Reconnect for Payment (Reconn), Start Service (Start) , Start/Stop Service (Start/Stop), Stop Service (Stop), Reread (Reread).
- The system uses the **Customer Event** to select the appropriate field activity type(s) from the field activity type profile associated with the service points linked to the service agreement.

WARNING:

The field activity types are NOT specified directly on the severance event type. Why? Because each service point linked to the service agreement being severed could necessitate a different type of field activity. Therefore, the system uses the type of service point, its state (e.g., connected, meter off) and the desired customer event (e.g., Disconnect Warning, Cut For Non-Payment) to determine which field activity type(s) to generate. Refer to [Setting Up Field Activity Type Profiles](#) for how to set up the specific disconnect field activity types for your various types of service points.

The Expire Service Agreement severance event type causes the service agreement to be moved to the pending stop state (it also populates the service agreement's stop date). Refer to [Finalizing Pending Stops](#) for how the system will eventually stop the service agreement (and then final bill it). There are no parameters for this type of event.

NOTE:

Cut for non-payment field activities are special. The Expire Service Agreement severance event type also makes any "cut for non-payment" field activities created by earlier severance events available to stop service. Specifically, it changes the linkage type of the field activities from Severance Activity to Stop Activity. You can see a service agreement's service points' field activities and their respective linkage type on [Service Agreement - Service Point](#).

Enter a **Long Description** to fully describe the severance event type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SEV_EVT_TYPE](#).

Setting Up Severance Process Templates

Severance process templates define the severance events that will be executed when a service agreement is severed. Open **Admin > Credit & Collection > Severance Process Template > Add** to define your severance process templates.

Description of Page

Enter a unique **Severance Process Template** and **Description** for the severance process template.

If severance processes of this type should be automatically canceled when the customer pays the collection amount on the severance, turn on **Auto Cancel**. This switch would typically only be turned off for severance processes used to reconnect a cut service because you don't want such a reconnection process to be canceled when a payment is made. Refer to [Designing Your Reconnection Procedures](#) for more information.

In addition to turning on the **Auto Cancel** switch, specify a **Cancel Criteria Algorithm** if your organization allows a severance process to be cancelled regardless of the debt associated with all service agreements in the debt class. In other words, if your cancel criteria are based on the debt associated with ALL service agreements in a debt class DO NOT SPECIFY THIS ALGORITHM.

If the **Cancel Criteria Algorithm** is specified, it is executed when a credit is posted to the service agreement associated with a severance process. This algorithm will indicate if the service agreement no longer has debt that warrants a severance process. Refer to [How Are Severance Processes Cancelled](#) for more information. If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that cancels a severance process if the service agreement's debt so warrants. Click [here](#) to see the algorithm types available for this plug-in spot.

If you wish to perform any special processes after a severance process is canceled, specify a **Post Cancel Algorithm**. This can be used to start a reconnection in case the severance process was canceled too late to stop the disconnection. If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to [Setting Up Algorithms](#)). On this algorithm, reference an Algorithm Type that cancels a severance process if the service agreement's debt so warrants. Click [here](#) to see the algorithm types available for this plug-in spot.

When a service agreement is to be severed due to non-payment, the system creates a severance process and links to it one or more severance events based on the **Event Types** entered here. The information in the scroll defines these events and the date on which they will be triggered. The following fields are required for each event:

Event Sequence controls the order in which the severance event types appear under the severance process template. The sequence number is system-assigned and cannot be changed. If you have to insert a severance event type between two existing templates, you'll have to remove the latter events, insert the new event, and then re-specify the removed events.

Severance Event Type Specify the type of severance event to be generated.

Dependent On Other Events Turn this indicator on if the trigger date of the event can only be determined after earlier events are complete. For example, you would turn this switch on for the event that initiates the field activity to disconnect service. Why? Because you only want to disconnect service after the preceding event that warned the client of impending disconnection is complete.

Days After Prev Response Specify the number of days after the completion / cancellation of the dependent events on which the severance event will be triggered. If this event is not dependent on the completion of other events, this field contains the number of days after the creation of the severance process that the related severance event will be triggered. Refer to [Severance Event Dependencies and Trigger Date](#) and [Calendar vs Work Days](#) for a description of how the system uses this information to set the trigger date on the respective severance events.

When the **Dependent On Other Events** switch is on, use the grid to define the events on which this event is dependent. If multiple events are specified in the grid, all such events must be completed or cancelled before the event will be triggered.

Sequence is system-assigned and cannot be specified or changed.

Dependent On Sequence Specify the sequence number of the severance event on which the above severance event depends.

FASTPATH:

For more information about severance event types, see [Setting Up Severance Event Types](#). For more information about trigger dates, see [Severance Event Dependencies & Trigger Date](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SEV_PROC_TMP](#).

Designing Your Write-Off Procedures

The design of your write-off procedures is relatively straightforward. Simply follow the instructions in the following topics.

Designing Your Write-Off Debt Classes

Multiple write-off debt classes are needed when you have different write-off procedures for different types of service agreements. If all service agreement debt is written-off the same way, then you'll have just one write-off debt class (call it Generic). However, if you're like many organizations, you will have multiple write-off debt classes. The following points will help you understand why:

- If you bill for 3 rd parties, you probably have different write-off debt classes for the 3 rd party service agreements. Why? Because you probably treat 3 rd party uncollectable debt differently from your own debt.
- You will need a separate write-off debt class for service agreements whose debt cannot be written off. Why? Because there is a switch on the write-off debt class control table that controls if service agreements in the write-off debt class are eligible for write-off processing. Given that you will have some service agreements that hold debt that aren't eligible for write-off processing (e.g., service agreements that hold written-off debt and service agreements that overpayments), you will need at least one other write-off debt class.
- If you use the system to calculate charges for your organization's company usage, you'll need another write-off debt class (we refer to it as the "N/A" write-off debt class below). Why? Because all service agreements must have a write-off debt class, even those that will never have debt.

SA's Write-Off Debt Class	Account's Collection Class	Account's Collection Class
Normal W/O		
N/A		

Designing Write-Off Controls

Set up a matrix using the collection classes you designed when you were designing your collection procedures ([Designing Your Collection Procedures](#)).

SA's Write-Off Debt Class	Account's Collection Class	Account's Collection Class
	Residential	Commercial/Industrial
Normal Write Off		
N/A		

Each cell should have a "write-off control" that defines what to do when the system detects finalized debt that hasn't been paid. This is true even of the "N/A" write-off debt class. Why? Because you may want the system to write-down these stopped SAs when they have a small balance. For example, if you have a write-off service agreement that subsequently receives a partial payment that leaves a small amount owing, you probably want the system to generate a write-down adjustment (so that the write-off service agreement will close). We'll initially fill in the matrix for the "N/A" write-off debt class.

SA's Write-Off Debt Class	Account's Collection Class	Account's Collection Class
	Residential	Commercial/Industrial
Normal Write Off		
N/A	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment.	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment.

NOTE:

If the Write Off Monitor encounters debt associated with a non-defined collection class and write-off debt class, it will issue an error.

For each cell that isn't designated as N/A, you need to answer the following questions:

- Are you allowed to transfer debt to other non-closed service agreements linked to the account? If so, you need to define the algorithm used to do such. Refer to [Setting Up Write-off Control](#) for more information about this algorithm.
- Are you allowed to write-down small amounts of debt (or small credits)? If so, you need to define the algorithm used to do such. Refer to [Setting Up Write-off Control](#) for more information about this algorithm.
- Should you refund credit balances with a check? If so, you need to define the algorithm to do such. Refer to [Setting Up Write-off Control](#) for more information about this algorithm.
- If debt remains after doing the above, how do you write it off (e.g., do you first refer the debt to a collection agency and only write it off after waiting 30 days)?

We'll fill in the above matrix with our assumptions:

SA's Write-Off Debt Class	Account's Collection Class	Account's Collection Class
	Residential	Commercial/Industrial
Normal Write-Off	Attempt to reduce the SA's balance to zero using the following methods: Attempt to transfer debt to another active service agreement linked to the account. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment. If the debt is <= \$-1, create an A/P adjustment to refund the credit to the customer. If debt remains: Highest priority: If customer has a non-cash deposit, create the non-cash deposit write-off process. Otherwise, create the default write-off process for residential debt.	Attempt to reduce the SA's balance to zero using the following methods: Attempt to transfer debt to another active service agreement linked to the account. If the debt / credit is < \$10 and > \$-10, write down the debt using a write-down adjustment. If the debt / credit is <= \$-10, create an A/P adjustment to refund the credit to the customer. If debt remains: Highest priority: If customer has a non-cash deposit, create the non-cash deposit write-off process. Otherwise, create the default write-off process for commercial debt.
N/A	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment. Because this debt class isn't eligible for further write-off processing, criteria used to process debt are not necessary.	Attempt to reduce the SA's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < \$10 and > \$-1, write down the debt using a write-down adjustment. Because this debt class isn't eligible for further write-off processing, criteria used to process debt are not necessary.

We can now use the information in the above matrix to design the necessary Write Off Process Templates and Write Off Event Types.

Designing Write Off Process Templates & Write Off Event Types

The following table shows the write-off process templates referenced in the previous section's matrix. Adjacent to each process are its events and an indication of when they are triggered.

Write Off Process Template	Write Off Event Type	Triggered X Days From Start Of Write Off Process
Residential	Refer to collection agency	0
	Letter notifying customer of referral	0
	Cancel collection agency referral	60
	Write off	60
Non-Cash Deposit Exists	To Do for non-cash deposit redemption	0
	To Do to highlight unpaid SA(s) still exist (and they will be reconsidered for write-off processing)	10
Commercial	Refer to collection agency	0
	Letter notifying customer of referral	0
	To Do to check up on collection agency's efforts	30
	Cancel collection agency referral	60
	Write off	60

If we extract each unique event type from the above table, we end up with the following:

Write Off Event Type	Event Type
Notification of write-off referral	Send letter - Debt referred to a collection agency
Refer to collection agency	Refer to collection agency
Cancel collection agency referral	Cancel collection agency referral
Write off	Write off
To Do for non-cash deposit redemption	Generate To Do - Redeem non-cash deposit
To Do to highlight unpaid SA(s) still exist	Generate To Do - SA(s) linked to a non-cash deposit remain unpaid (and will be reconsidered for write-off processing)
To Do to check up on collection agency's efforts	Generate To Do - Check up on collection agency's efforts

And now you're ready to set up your write-off procedures.

Setting Up Write-Off Procedures

In the previous section, [Designing Your Write-Off Procedures](#), we presented a case study that illustrated a mythical organization's write off procedures. In this section, we'll explain how to set up the control tables to implement these procedures:

Setting Up Write Off Debt Classes

Every SA type has a write-off debt class. This class is one of several fields that control the write off criteria applied to the service agreement's debt. Select **Admin > Credit & Collection > Write Off Debt Class** to define your debt classes.

FASTPATH:

For more information about debt classes, see [Designing Your Write-Off Debt Classes](#).

Panel controls

To modify a write-off debt class, simply move to a field and change its value. To add a new write-off debt class, click + to insert a row, then fill in the information for each field. The following fields display:

Write Off Debt Class Code The unique identifier of the write off debt class.

Eligible for Write Off Indicates if service agreements belonging to this write off debt class are eligible for write-off processing. This should only be turned off if this debt cannot be written off, e.g., write off debt.

Description The description of the write off debt class.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_WO_DEBT_CL](#).

Setting Up Write Off Event Types

Write-off event types control what is done by a given write-off event. Select **Admin > Write Off Event Type > Add** to define your write-off event types.

Description of Page

Enter a unique **Write Off Event Type Code** and **Description** for the write-off event type.

Enter the **Write Off Event Type**. Permissible values are: Affect Credit Rating/Cash-Only , Cancel Agency Referral, Generic Algorithm, Refer to Agency , Send Letter, Create To Do Entry, Write Off using Distrib Code , Write Off using SA Type. The following discussion describes the parameters that must be defined for each type of write-off event.

The Affect Credit Rating/Cash-Only write-off event type causes a credit rating demerit record to be linked to the account. This record is constructed using the following **Parameters**:

- Use **Affect Credit Rating By** to define this event's affect on the account's credit rating. This should be a negative number. An account's credit rating is equal to the start credit rating amount defined on the installation record plus the sum of credit rating demerits that are currently in effect. When an account's credit rating is less than the credit rating threshold defined on the installation record, the account's credit rating is displayed as an alert on Control Central.
- Use **Affect Cash-Only Score By** to define this event's affect on the account's cash-only score. This should be a positive number. When an account's cash-only score exceeds the cash-only threshold score defined on the installation record, the account is flagged as cash-only during payment processing and on Control Central.
- Use **Months Affecting Credit Rating** to define the length of time the demerit remains in effect. This information is used to define the effective period of the credit rating demerit record.

FASTPATH:

For more information, refer to [Account - Credit Rating](#).

The Cancel Agency Referral event type will cancel previous collection agency referrals. No parameters are needed for this type of event.

The Generic Algorithm write-off event type causes the algorithm defined in the **Generic Algorithm** to be executed. You use this type of algorithm when the standard types of write off events won't do what you need done. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines the algorithm that will be called when events of this type are activated. Click [here](#) to see the algorithm types available for this plug-in spot.

The Refer to Agency event type will refer the debt associated with the process' SAs to a collection agency. You must supply the **Agency Selection Algorithm** that is used to determine the collection agency associated with the referral. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines the collection agency to which bad debt should be referred. Click [here](#) to see the algorithm types available for this plug-in spot.

NOTE:

Letters. You must set up a customer contact type for each type of letter you generate. You specify the necessary customer contact type on the write off event type. Refer to [Setting Up Letter Templates](#) for more information.

The Send Letter write-off event type causes a customer contact to be generated that, in turn, generates a letter. Enter the following parameters for this type of event:

- Select the **Contact Class** used to categorize the customer contact.
- Use **Contact Type** to define the type of customer contact to create. The type of customer contact controls the type of letter that is generated.

The Create To Do Entry write-off event type causes a To Do entry to be issued. Refer to [The Big Picture of To Do Entries](#) for more information about To Do entries (refer to the To Do type TD-WOEV for the type of To Do entry that's created).

The Write Off using Distrib Code event type causes bad debt to be written off in accordance with the distribution codes associated with the financial transactions that caused the debt in the first place. Use this method if, for example, you want to write-off revenue differently than you write-off liabilities. When this type of event is activated, the system accumulates the distribution codes from GL details associated with recent financial transactions linked to each write-off service agreement. When the system has accumulated enough distribution codes (i.e., where the amount associated with the distribution code equals or exceeds the amount to write off), the debt will be transferred to a new or existing write-off service agreements. The type of service agreements to which the debt is transferred is defined on the distribution codes. Refer to [Setting Up Distribution Codes](#) for more information.

The Write Off using SA Type event type causes all debt associated with the process' SAs to be transferred to a write-off service agreement linked to the account. Enter the following **Parameters** for this type of event:

- **CIS Division / SA Type** is the type of write-off service agreement to which the debt will be transferred. Note well,
 - The system will reuse an existing service agreement if an active SA of this type is already linked to the account; otherwise the system will create a new service agreement of this type.
 - The adjustment type used to set the offending service agreement's current balance equal to its payoff balance is defined on the write-offable SA Type. Refer to [SA Type - Main Information](#) for more information.
 - The adjustment type used to transfer the delinquent debt to the write-off service agreement is defined on the write off SA type. Refer to [SA Type - Detail](#) for more information.

Enter a **Comment** to fully describe the write-off event type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_WO_EVT_TYP](#).

Setting Up Write Off Process Templates

Write-off process templates define the write-off events that will be executed when a write-off criteria rule is violated. Select **Admin > Credit & Collection > Write Off Process Template > Add** to define your write-off process templates.

Description of Page

Enter a unique **Write Off Process Template** code and **Description** for the write-off process template.

The rows in the following grid define the events that will be created when a write off process is created using this template. The following fields display:

Event Sequence controls the order in which the write-off event is executed. The sequence number is system assigned and cannot be changed. If you need to insert a write-off event between two existing events, you must remove the latter events, insert the new event, and then re-enter the removed events.

Write-off Event Type Code Specify the type of write-off event to be generated. The event type's description is displayed adjacent.

Days After Process Creation Specify the number of days after the creation of the write-off process that the related write-off event will be triggered. Refer to [Calendar vs Work Days](#) for a description of how this system uses this information to set the trigger date on the respective write-off events.

FASTPATH:

For more information about write-off event types, see [Setting Up Write Off Event Types](#). For more information about trigger dates, see [Write-off Event Trigger Date](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_WO_PROC_TMPL](#).

Setting Up Write Off Control

Write-off controls define how the system handles finalized, unpaid debt belonging to a given collection class and write off debt class.

Write Off Control - Main

Select **Admin > Consumption > Write Off Control > Add** to define basic information about a write-off control. After entering basic information, navigate to the **Criteria** tab to define the type of write-off process to start when given criteria are met.

FASTPATH:

For more information about write-off control, refer to [Designing Write-Off Controls](#).

Panel controls

Enter a **Write Off Control** code and **Description** for the write-off control (WOC).

Enter the **Collection Class** to which the WOC applies.

Enter the **Write Off Debt Class Code** to which the WOC applies.

Enter general **Comments** to further describe the WOC.

Define the **Synch All Algorithm** used by the system to generate adjustments that cause current balance to equal payoff balance on the service agreements to be written off. This type of algorithm is issued before you actually start a write-off process as current balance is meaningless at write-off time (the customer owes you the payoff balance). If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that synchronizes current and payoff balances. Click [here](#) to see the algorithm types available for this plug-in spot.

Define the **Debt Transfer Algorithm** used by the system when it attempts to transfer the unpaid debt to another active service agreement linked to the stopped SA's account. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that transfers unpaid balances. Click [here](#) to see the algorithm types available for this plug-in spot.

Define the **Write Down Algorithm** used by the system when it attempts to write-down small debt and/or credit balances. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that writes down small amounts. Click [here](#) to see the algorithm types available for this plug-in spot.

Define the **Credit Refund Algorithm** used by the system when it refunds a credit balance to a customer. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that refunds credit balances. Click [here](#) to see the algorithm types available for this plug-in spot.

Write Off Control - Criteria

Select **Admin > Consumption > Write Off Control > Search** and navigate to the **Criteria** page to define the type of write-off process to start when given criteria are met.

FASTPATH:

The following information is not intuitively obvious. Refer to [Designing Write-Off Controls](#) for more information.

Panel controls

The information in the grid defines the write-off process to be executed for debt belonging to the previously defined collection class and write off debt class. The type of write-off process may differ depending on some condition. For example, you may have a different write-off process if the customer has a non-cash deposit. You must have at least one entry in this collection otherwise the system will not start a write-off process. This entry should have the lowest priority code and should reference a **Write Off Criteria Algorithm** that references the WO CRIT DFLT the algorithm type.

The following fields display:

Priority controls the order in which the criteria will be checked by the Write Off Monitor (higher priorities are checked before lower priorities). The first criteria algorithm that is met (i.e., returns a value of *True*) will cause the associated write-off process to be initiated.

NOTE:

The values for this field are customizable using the Lookup table. This field name is CRIT_PRIO_FLG. Be aware that this field is used for multiple tables: [Collection Class Control](#), [Severance Criteria](#), [Write Off Control](#) and [Workflow Process Profiles](#).

Write Off Criteria Algorithm The Write Off Monitor checks if the condition defined by the W/O Condition Algorithm applies to the account whose debt is being analyzed. If a condition is met, a write-off process is created using the associated write-off process template.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
 - On this algorithm, reference an Algorithm Type that determines if a customer's bad debt should be processed using the associated **Write Off Process Template**. Click [here](#) to see the algorithm types available for this plug-in spot.
-

IMPORTANT:

You must have at least one entry in this grid otherwise the system will not start a write-off process. This entry should have the lowest priority code and should reference a **W/O Criteria Algorithm** that references the [WO CRIT DFLT](#) algorithm type.

Write Off Process Template If the Write Off Monitor determines the condition defined by the w/o condition algorithm applies, a write-off process is created using the associated write-off process template.

Where Used

Write-off controls contain the data that controls the Write Off Monitor. Refer to [How Does The Write-Off Monitor Work?](#) for more information.

Setting Up Collection Agencies

You must set up a collection agency for each such organization to which you refer delinquent debt. To define a collection agency, select **Admin > Credit & Collection > Collection Agency**.

Description of Page

Enter an easily recognizable **Collection Agency** code and **Description** for each collection agency.

A collection agency must be associated with a Person. Choose the **Person ID** of the organization from the prompt.

FASTPATH:

Information about how to set up persons is discussed in [Maintaining Persons](#).

Turn on the **Active** switch if the collection agency is actively receiving referrals.

Specify the **Batch Control** that's used to route new and cancelled referrals to the collection agency. The batch control's description is displayed adjacent.

Where Used

Collection agencies get assigned to collection agency referrals when the collection agency referral background process executes. Refer to [How Do Collection Agency Referrals Work?](#) for more information.

Setting Up Feature Configuration

You must set up a [Feature Configuration](#) if you use the [champion / challenger](#) functionality.

The following describes the various **Option Types** that must be defined:

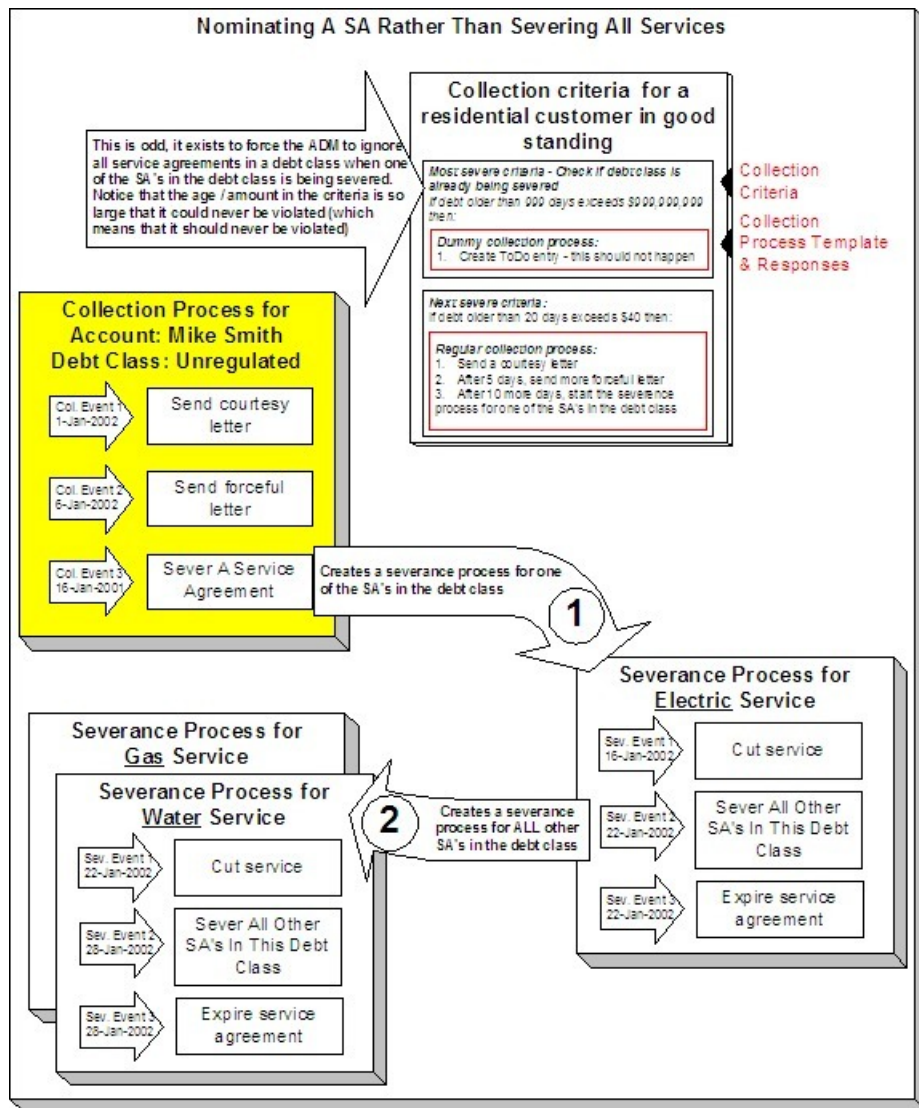
Champion Template\$Challenger Template\$Percentage(1-100). You need only set up options of this type if your implementation implements **Champion / Challenger** functionality. Options of this type are entered in the format A\$B\$nnn where A is the collection process template of the champion template, B is the collection process template of the challenger template, and C is the percent of the time that the system should create the challenger template. The collection monitor uses this option to override the champion collection process template X% of the time with the challenger template. You may enter any number of these options (but only one per Champion Template).

How To

The contents of this section describe how to set up various credit and collections scenarios.

How To Nominate A Single Service To Sever (Rather Than Sever Everything That's In Arrears)

Some organizations that offer multiple services do not sever all services when the customer falls into arrears. Rather, they nominate a single service agreement to sever in the hopes that the lack of service will cause the customer to remit payment. The following diagram illustrates the control tables values required to implement this type of requirement.



The following important concepts are illustrated above:

- The collection process's last event does NOT sever all services. Rather, it calls an algorithm that selects a single service to sever. A base package algorithm allows you to define the primary service to sever and a secondary service to sever (if the customer does not have the primary service). If you sold electricity and gas, you would probably define the primary service as electricity and the secondary as gas (because electricity is easier to cutoff than gas).
- The severance process that is started for the primary service cuts service. If the customer doesn't remit what is owed, the second severance event calls an algorithm that severs all other service agreements in the debt class.
- Because you are nominating a single service to sever, you must set up a special value in collection class control to force the ADM to ignore all service agreements in a debt class when one of the SAs in the debt class is being severed. Notice that the age / amount in the criteria is so large that it could never be violated.
- In addition, because the entire debt class must no longer be in arrears to stop the collection and severance processes, you must plug-in the appropriate collection process and severance process cancellation criteria on the debt class. Refer to [How Are Collection Processes Cancelled](#) and [How Are Severance Processes Cancelled](#) for more information about how these algorithms are used. Also note, you do not need service agreement cancellation criteria defined on your collection process templates and severance process templates (because cancellation is controlled at the debt class level).

Defining Meter & Item Options

Located at a premise's service points are the various meters, items and equipment that regulate and measure consumption. Before you can define meters, items and equipment you must set up the control tables defined in this section.

FASTPATH:

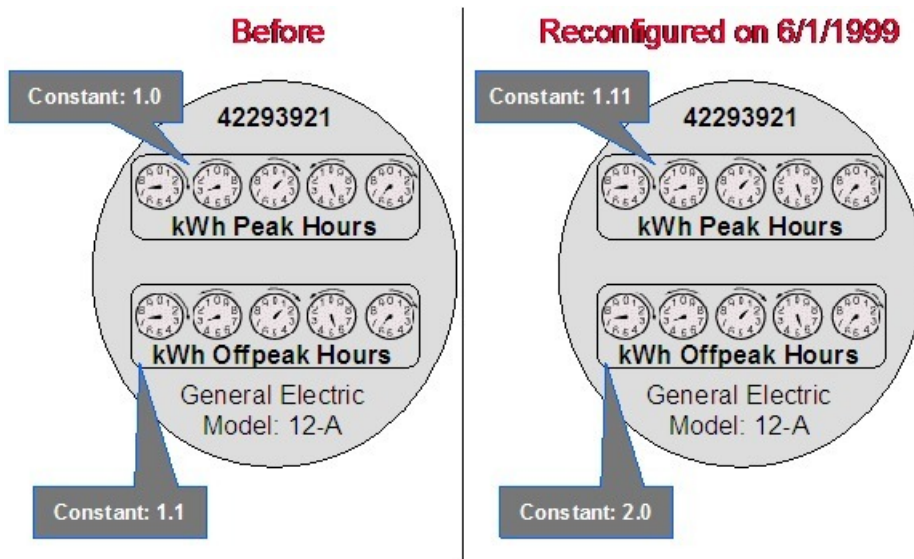
Refer to [An Illustration Of A Premise](#) for an illustration of how meters, items and equipment are related to the service points.

The Big Picture of Meters, Items and Equipment

The topics in this section provide background information about meters, items and equipment.

The Structure Of A Meter

A customer's consumption is measured using a meter. What a meter measures and how it measures it can change over time. The following example illustrates a simple meter before and after a reconfiguration.



The system maintains how a meter looks over time so that it can reproduce bills using historical consumption. Information about this meter is recorded in the following tables:

- Every meter has a single **meter** record that contains information about the meter that doesn't change over time. For example, its meter number and manufacturer.
- A **meter configuration** record is required whenever something changes about what the meter measures. The meter shown above has two configurations - the original and the one effective on 1-June-1999.
- Every meter configuration contains one or more **registers**. Each register references the unit being measured and how the measured quantity is manipulated before it is billed.

FASTPATH:

Information about how to set up meters is discussed in [Maintaining Meters](#).

Items Are Used For Other Devices Associated With A Customer's Service

In addition to meters, there are many other devices that can be involved with a customer's service. We refer to these other devices as "items".

We purposefully use the ambiguous term *item* because items are used for many different devices including lamps, poles, current transformers, backflow devices, pulse initiators, etc. Refer to the diagram in [An Illustration Of A Premise](#) for an example of the various items that could be associated with a customer's service.

The topics in this section provide more information about items.

Equipment versus Badged Items

This is a rather confusing subject, but there are two notional types of items:

- **Equipment.** An item that is considered to be "equipment" is a physical device that regulates consumption; it does NOT measure consumption. You would only define equipment if it is of interest to your organization. For example, if your organization periodically tests the pulse initiators associated with your meters, you will need to set up items for each pulse initiator and link them to their respective meters. Equipment can be linked to either a service point (e.g., a current transformer, a backflow device), a meter (e.g., a pulse initiator), or an item (e.g., the components of an installation).
- **Non Equipment.** An item that is not considered to be "equipment" is a physical device that does NOT measure consumption, but impacts billing in some way (i.e., there are charges in your rates based on the number and type of items installed at a service point). Examples include street lights, light poles, and security cameras. These types of items are related to service points. Refer to [Chargeable Items Must Be Associated With Service Points](#) for information about the two types of items that may be linked to service points.

The topics that appear below provide more information about both types of items.

Meters May Have Equipment

If you have physical devices that are intrinsically associated with meters (but aren't meters), you can set up an item for each such device. For example, if you have pulse initiators linked to your meters, you would set up an item for each pulse initiator. After creating these items, you would update each meter's collection of equipment to include the respective pulse initiator.

Items May Have Equipment

If you have physical devices that are intrinsically associated with other non-metered devices, you can set up an item for each such device. For example, you could set up an item for an "installation" (an installation is a group of devices that work together to regulate electric consumption) and link to it the individual items that do the work. To do this, you would set up items for each individual piece of equipment in the installation. After creating these items, you would update the collection of equipment associated with the installation's item to include each piece of equipment.

Service Points May Have Equipment

If you have physical devices that don't have line item charges but are intrinsically associated with a service point, you can set up an item for each such device. For example, if you have voltage regulators linked to your service points, you would set up an item for each voltage regulator. After creating these items, you would update each service point's collection of equipment to include the respective voltage regulator.

Chargeable Items Must Be Associated With Service Points

If there are charges in your rates based on the number and type of items installed at a service point, you can use either of the following techniques:

- If the item is badged (i.e., uniquely identified), you would set up an item and link it to an **item-based** service point. An item-based service point may have zero or one item installed at any instant in time. Over time, an item-based service point may have many badged items installed and removed. Refer to [Installing / Removing An Item](#) for more information.
- If the item does not have a unique identifier, you do NOT have to set up phony items for it. Rather, you can use the system's multi-item functionality and simply define the number of each TYPE of item that is installed at a service point. Refer to [Service Points With Multiple Items](#) for an illustration of such a service point.

Start / Stop and Items

When a customer start or stops service at an **item-based** service point, it takes into account whether an item is currently installed at the SP and whether the item is on or off. Start / Stop does not concern itself with equipment.

Billing and Items

It's important to note that billing ignores equipment relationships when it constructs the snapshot of number and types of items associated with a service agreement's service points. In other words, billing constructs the item snapshot as follows:

- It inserts an entry into the item snapshot for every item type referenced in the service agreement's service points' multi-item collection.
- It inserts an entry into the item snapshot for every badged item linked to service points with an SP type of Item.

FASTPATH:

Refer to [Item Snapshot](#) for more information.

NOTE:

Equipment and billing. Be aware that the only way equipment can impact billing would be if you develop a pre-processing calculation group that analyzed the equipment associated with a service point (directly or indirectly via the meters and items) and manipulated billed consumption accordingly. Refer to [All Calculation Rule-Based Rates Share a Common Structure](#) for more information about pre-processing calculation groups.

Generic Equipment

Some companies do not badge their equipment, but they still need to keep track of the type of equipment that is linked to service points, meters and badged-items. For example,

- You may want to keep track of the type of modem linked to a meter, but you don't care about the individual modem or pulse-initiator

- You may want to keep track of the type of backflow device linked to a water meter

NOTE:

Why is generic equipment recorded in the system? Generic equipment gets linked to SP's / meters / items when the type of equipment linked to an SP / meter / item affects bill calculations. For example, some water companies charge extra if a backflow device is present at a service point.

If you encounter the need for generic equipment, do the following:

- Create an Item Type for each type of unbadged equipment. On this item type, indicate that items of this type may be linked to multiple SP's / meters / items. Note: we recommend suffixing the description of the item type with the word "generic".
- Update the relevant meter type(s) / SP type(s) / item type(s) to indicate that they may use equipment associated with the generic item type.
- Create one (1) item for each type of unbadged equipment. Note: we recommend suffixing the description of the item type with the word "generic".
- Link the generic item to the relevant SP's / meters / items.

Setting Up Meter Options

This section describes tables that must be set up before you can define meters.

Setting Up Meter Configuration Types

Every meter configuration must reference a meter configuration type. The meter configuration type indicates the valid (required or optional) unit of measure and time of use registers for the configuration.

Navigate to **Admin > Meter Configuration Type > Add** to set up valid meter configuration types.

Description of Page

Enter an easily recognizable **Meter Configuration Type** and **Description** for the meter configuration type.

Choose the **Service Type** associated with the meter configuration type. Refer to [Setting Up Service Types](#) for more information.

Use **Prepaid Meter** to indicate whether or not the meter configuration is used to record prepaid usage.

If you have set up a **TOU Group** to indicate your collection of time of use codes for the meter configuration, enter it here.

Use the **Default TOU Registers** button to have the system build the collection of registers for you based on the TOU codes linked to the TOU group. Refer to [Setting Up TOU Groups](#) for more information

In the grid, specify the attributes of the collection of valid registers. If the register is **Interval**, put a check in the checkbox. For any register, indicate the valid **UOM**. For interval registers, indicate the **Interval Register Type**. If a **TOU** is applicable for the register, enter the TOU code. Finally, specify a **Validation** for each register. The register pair can be Optional or Required.

NOTE:

In certain markets, registers with the same **UOM** and **TOU** combination or, for **Interval** meters, the same **UOM** and **Interval Register Type** combination may need to be defined under a single meter configuration. If your implementation is in one of these markets, add the Allow Duplicate UOM/TOU Combination **Option Type** on the Meter Management Options [Feature Configuration](#) and set its value to Y .

Attributes related to interval registers may not appear. These attributes are suppressed if Meter Data Management module is [turned off](#).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MTR_CONFIG_TY](#).

Setting Up Meter Types

Every meter references a meter type. The meter type defines the type of service and common characteristics shared by its meters. The meter type also controls the characteristics that may be specified on meters of a given type.

NOTE:

When a new meter type is added. After adding a new meter type, you must define the SP types at which meters of this type can be installed.

The topics in this section describe how to maintain your meter types.

Meter Type - Main

To define a meter type, open **Admin > Meter Type > Add**.

Description of Page

Enter an easily recognizable **Meter Type** and **Description** for the meter type.

Choose the **Service Type** associated with all meters of this type. Refer to [Setting Up Service Types](#) for more information.

Turn on **Allow Duplicate Meter Badges** if more than one meter of this type may have the same badge number.

Turn on **Prepaid-Capable** if meters of this type are used to record prepaid usage.

Turn on **Allow Interval Registers** if meters of this type may contain interval registers. Refer to [The Big Picture of Raw Data Collection and Aggregation](#) for more information.

Indicate in **Track Location** whether you Track or Do Not Track the location of meters of this type. Refer to [The Big Picture of Asset Inventory](#) for more information.

Use the **Characteristic Values** collection to define **Characteristic Types** and their respective **Characteristic Values** and **Adhoc Values** to describe characteristics common to all meters of this type.

NOTE:

You can only choose characteristic types defined as permissible on the meter type record.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MTR_TYPE](#).

Meter Type - Meter Characteristics

To define characteristics that can be defined for meters of a given type, open **Admin > Meter Type > Search** and then navigate to the **Meter Characteristic** page.

Description of Page

Use the **Meter Characteristics** collection to define characteristics that can be defined for meters of a given type. Turn on the **Required** switch if the **Characteristic Type** must be defined on meters of a given type. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

Meter Type - Equipment Types

Open **Admin > Device > Meter Type > Search** and then navigate to the **Equipment Types** page to define the types of equipment that can be linked to meters of a given type. Refer to [Equipment versus Badged Items](#) for more information about equipment.

Description of Page

Use the collection to define the item types of **Equipment** that can be linked to meters of this type.

NOTE:

Item types are being specified. There is no equipment type control table. Rather, items are used to define equipment and therefore you are actually defining item types rather than equipment types. Refer to [Equipment versus Badged Items](#) for more information about equipment.

Meter Type - Test Types

Open **Admin > Device > Meter Type > Search** and then navigate to the **Test Types** page to define the types of device tests that can be performed on meters of a given type. Refer to [The Big Picture Of Device Testing](#) for more information about device tests.

Description of Page

Use the **Device Test Type** collection to define the types of [device tests](#) that can be performed on meters of a given type.

Meter Type - Meter Configuration Type

Open **Admin > Device > Meter Type > Search** and then navigate to the **Meter Configuration Type** page to define the types of meter configurations that are allowed for this type of meter.

Description of Page

Use the **Meter Configuration Type** collection to define the types of meter configurations that can be used for this type of meter.

Setting Up Manufacturers & Their Models

When you set up a meter (or an item) you must define the manufacturer and model number of the meter / item. To define a manufacturer and its models, open **Admin > Manufacturer > Add**.

Description of Page

Enter an easily recognizable **Manufacturer** and **Description** for the manufacturer.

Enter a **Model** and **Description** for every model supplied by the manufacturer. Enter the **Service Type** with which the model is associated. Refer to [Setting Up Service Types](#) for more information.

If the model is associated with a meter, turn on **Use On Meter**. If the model is associated with an item, turn on **Use On Item**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MFG](#).

Setting Up Meter ID Types

A meter may have alternate ID's other than its badge number. If so, it will reference one or more meter ID types (one for each alternate form of ID). To define meter ID types, open **Admin > Meter ID Type**.

Description of Page

Enter a unique **Meter ID Type** and **Description** for every meter ID type.

Where Used

A meter may have alternate ID's other than its badge number. If so, it will reference one or more meter ID types (one for each alternate form of ID). Refer to [Meter - Meter ID Information](#) for more information.

Setting Up Read Out Types

Every register has a read out type that defines how the register's measurements are physically displayed (e.g., circular dials, digital, n/a-electronic transmission, and so on). To define read out types, open **Admin > Device > Read Out Type**.

Description of Page

Enter a unique **Read Out Type** code and a **Description** for every read out type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_READ_OUT_TYP](#).

Setting Up Protocol Codes

Every register has a protocol code that defines the method used to record the register's measurements (e.g., visual, probe, AMR, modem). To define protocol codes, open **Admin > Device > Protocol**.

Description of Page

Enter a unique **Protocol** and **Description** for every protocol.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PROTOCOL](#).

Setting Up Unit Of Measure Codes

You must create a unit of measure (UOM) for:

- Every UOM that is measured by your meters' registers.
- Every UOM in which your rates' prices are expressed.
- Every UOM in which your items' estimated consumption is expressed.

NOTE:

Typically, the measured UOM is the same as the UOM in which the prices are expressed. However, there are instances where the UOM of the price is not measured by a meter's registers. Natural gas is a good example - the prices are expressed in an amount per therm, but the gas meters measure some volume (e.g., cubic feet, cubic meters).

FASTPATH:

For more information about UOM's, refer to [UOM versus TOU versus SQI](#).

To define unit of measure codes, open **Admin > General > Rates > Unit of Measure**.

Description of Page

The following fields display for each unit of measure:

UOM The unique identifier of the unit of measure.

Description The full description of the UOM.

Service Type The type of service (e.g., electric, gas, water, and so on.) associated with this UOM. This value controls the UOMs that may be referenced on meters belonging to a given service type. Refer to [Setting Up Service Types](#) for more information.

Decimal Positions The number of decimal positions that appear on bill lines that show consumption.

NOTE:

Suppression of trailing zeroes. If you wish to suppress trailing zeroes in the consumption that appears on bill lines, you must set up the Allow Zero Suppression Of Consumption In Bill Description option type on the Financial Transaction Options [Feature Configuration](#).

Allowed on Register Turn on this switch if the UOM can be referenced on a meter's register. A unit of measure may not be allowed on a register when it exists purely because there is a price in a rate expressed in this UOM. (For example, prices of natural gas are frequently expressed in therms, but it is rare to find a meter that measures gas in therms.)

Measures Peak Quantity Turn on this switch for UOMs that exist to record the peak amount of consumption, e.g., kW and kVar. Peak UOMs are treated differently by billing when determining the amount of consumption. They are also treated differently if rating has to prorate consumption.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_UOM](#).

Setting Up Time-Of-Use Codes

You will only create time of use (TOU) codes if your meters measure consumption in respect of broad time bands. For example, some electric meters measure consumption in respect of WHEN the power was used - peak period, off peak period, or partial peak period.

If you have "time of use" meters, then you must create TOU codes for:

- every TOU that is measured by your meters, and
 - every TOU in which your rates' prices are expressed.
-

NOTE:

Typically the measured TOU is the same as that in which the prices are expressed. However, there are instances where the TOU of the price is not measured directly by a meter's registers. A simple example is where the meter measures peak hours consumption and total consumption but there are prices in the rate for off peak consumption. In this situation, the

peak hours consumption would have to be subtracted from the total consumption to derive off peak consumption. In this situation, you would need TOU codes for both peak (measured) and off-peak (derived) periods.

FASTPATH:

For more information about TOU's, refer to [UOM versus TOU versus SQL](#).

Time-of-use codes will also be needed if your company offers interval billing with TOU mapping. Refer to [Designing Your Time of Use Options](#) for more information.

To define time-of-use codes, open **Admin > Rates > Time of Use**.

Description of Page

Enter a unique **TOU** code and **Description** for every time of use code.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TOU](#).

Setting Up TOU Groups

TOU Groups may be used to group together time of use codes, which are included in a meter configuration. It is also used in TOU mapping and pricing functionality.

FASTPATH:

Refer to [Grouping of TOU Codes](#) for more information about the TOU group's role in TOU mapping and pricing.

Open **Admin > TOU Group > Add** to define your TOU Groups.

Description of Page

Enter a unique **TOU Group** and **Description** for the TOU group.

Enter the Collection **Time of Use** codes. This is a list of time of use codes that define the time of use periods to be used for certain TOU maps. For each time of use code, you may indicate a **TOU Sequence** to indicate the relative order or relative priority of each TOU for the TOU group. Refer to [Grouping of TOU Codes](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TOU_GRP](#).

Setting Up Retirement Reasons

When you change a meter or item's status to Retired, you must supply a retirement reason. To define retirement reasons, open **Admin > Devuce > Retire Reason**.

Description of Page

Enter a **Retire ReasonCode** and **Description** for every retirement reason.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_RETIRE_RSN](#).

Setting Up Metered Service Point Options

This section describes tables that must be set up before you can define metered service points.

Defining Meter Location Codes

When you set up a metered service point you must define where the meter's service point is located on the property.

NOTE:

The meter location code is provided to meter readers and field workers to help locate the meter.

To define meter location codes, open **Admin > Meter Location**.

Description of Page

Enter a unique **Meter Location** and **Description** for every meter location.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MTR_LOC](#).

Setting Up Service Cycles And Routes

FASTPATH:

Refer to [Designing Service Cycles, Routes, And Schedules](#) for more information.

Setting Up Metered Premise Options

This section describes tables that must be set up before you can define premises.

Setting Up Meter Read Instructions

When you set up a premise you may define instructions to be supplied to the individuals who read the meters located at the premise. To define meter read instruction codes, open **Admin > Geographic > Meter Read Instruction**.

Description of Page

Enter a unique **Meter Read InstructionCode** and **Description** for every meter read instruction.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MR_INSTR](#).

Setting Up Meter Read Warnings

When you set up a premise you may define warnings to be supplied to the individuals who read the meters located at the premise. To define meter read warning codes, open **Admin > Geographic > Meter Read Warning**.

Description of Page

Enter a unique **Meter Read Warning** and a **Description** for every meter read warning.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MR_WARN](#).

Setting Up Consumption Estimation Parameters

The system estimates consumption under the following situations:

- At billing time, when "real" consumption cannot be computed due to the lack of a meter read, the system estimates how much the customer used assuming the following conditions are true:
 - The service agreement allows estimation.
 - The rate schedule allows estimation.
 - The bill cycle schedule (if one is being used) allows estimation.
 - The register is not a peak register (this is an artificial constraint, but most organizations do not estimate peak consumption). Note that the base product estimation algorithms may attempt to estimate peak registers.
 - The register is subtractive. This is because only subtractive registers are self-correcting. For example, assume we estimate a reading of 100 and the next real read is 102, the customer will only be charged for 2 units. But consider what would happen if we estimated a consumptive register, the consumption associated with the next real read will be billed in its entirety.
- When a meter read is added, the system verifies that the resultant consumption is congruous with historical consumption trends (i.e., high / low checks are performed).

This section describes the tables that must be set up to estimate consumption.

Estimating Consumption

How the system generates estimated consumption is up to you because a plug-in algorithm is called when it's time to estimate consumption. The identity of the plug-in algorithm is defined on the [trend area](#) of the premise at which the meter is installed.

The base package is supplied with two algorithm types that you may use to estimate consumption:

- The [TSMRE-LA](#) algorithm type first tries to estimate consumption using historical data for the account and service point. If this is unsuccessful, it uses trend data to estimate consumption.
- The [MR EST TREND](#) algorithm type uses trend data to estimate consumption.

If neither of these algorithms works for your organization, you will need to write a new algorithm and plug it in on your trend areas.

The following discussion explains how the sample estimation plug-ins work.

Estimating Using Historical Consumption

One of the sample estimation algorithms provided with the base package attempts to estimate consumption using historical data for the account and service point.

When you estimate consumption using historical data, you assume that the customer will have current consumption that is similar to their historical usage. For example, if you need to estimate consumption for the month of February of the current year, you could assume that the customer has similar consumption during the month of February of the previous year or that they have similar consumption to the previous month.

The base package algorithm type that uses a customer's historical data to estimate consumption is called [TSMRE-LA](#) (three-step meter read estimation). It makes three attempts to estimate consumption stopping at the first successful attempt.

First, it tries to estimate consumption using a bill segment for the service point from the previous year that covers a similar period to the one being estimated. The year-old bill segment can be used for estimation if it is for the same account AND the bill segment's end read is not a system estimate AND there are enough days in the bill segment for estimation (as defined by the algorithm).

If the year-old bill segment cannot be used for estimation, the system attempts to use the bill segment for the account and service point that immediately precedes the start date of the estimation period. Again, this bill segment must not have an estimated end read and it must have a sufficient number of days for estimation.

Once a historical bill segment that can be used for estimation is located (using one of the above methods), the estimation occurs as follows:

- If the register is non-peak, the system calculates the average amount of consumption per day on the historical bill segment and multiplies this by the number of days in the current estimation period. For example, if the customer averaged 30.350877 kWh per day in the historical bill period and there are 30 days in the current period, the estimated consumption is: 911 kWh.
- If the register is peak, the estimation amount is the same as the peak amount on the historical bill segment.

If the system is unsuccessful in estimating consumption using the historical data for the previous year or the previous bill period, the system estimates consumption [using trends](#).

Estimating Using Trends

WARNING:

The topics that appear below explain how the [MR EST TREND](#) sample estimation algorithm works. Another algorithm is also available in the base package. It attempts to estimate consumption using the customer's historical consumption; if historical consumption does not exist, it estimates using the logic described below (refer to [Estimating Using Historical Consumption](#) for the information about this algorithm).

The Theory Behind Consumption Estimation Using Trends

The following discussion explains how the sample estimation plug-ins use trend data. If you decide to create your own version of one of these plug-ins, you still have to set up trend areas and trend classes as these are required fields on premises and SP types.

The standard estimation plug-ins assume that if a customer historically used, say, twice as much as customers of a similar profile, then the customer should use twice as much in the current period. Therefore, to estimate consumption for any period of time the system needs to know:

- A. How much the specific customer used in the previous consumption period.
- B. How much the average customer in the customer's trend profile used in the previous consumption period.
- C. How much the average customer in the customer's trend profile used in the period being estimated.

Once the above are known, estimated consumption for the specific customer equals $(A/B) * C$.

Knowing how much the specific customer used in the previous consumption period simply involves looking at the customer's previous readings (or reading, in the case of consumptive meters).

Knowing how much the average customer used requires consumption trend data. Consumption trend information is continuously updated behind-the-scenes using meter reads. A consumption trend is identified by a unique combination of:

- **Trend area.** Trend areas are used to differentiate consumption trends in different geographic areas. If your service territory doesn't have appreciable differences based on geography, then you will have just one trend area.
- **Trend class.** Trend classes are used to differentiate consumption trends based on the type of service and the type of property. At a minimum, you would probably have a trend class to differentiate between residential, commercial, and industrial usage.
- **Unit of Measure and Time of Use.** All consumption in the system is identified using a unit of measure code and, optionally, a time-of-use code.

FASTPATH:

For more information about unit of measure codes, see [Setting Up Unit Of Measure Codes](#). For more information about time-of-use codes, see [Setting Up Time-Of-Use Codes](#). Trend areas, trend classes, and consumption trends are described later in this section.

NOTE:

Bottom line. A separate consumption trend is maintained behind-the-scenes for every combination of trend area, trend class, unit of measure and time of use.

Consumptive Register Where UOM Does Not Measure Peak Quantity

The algorithm used to estimate consumption for a consumptive, non peak unit of measure is identical to [Subtractive Register Where UOM Doesn't Measure Peak Quantity](#); the only difference is that we don't have to find the previous, previous read in order to determine the customer's usage in the previous period.

Subtractive Register Where UOM Does Not Measure Peak Quantity

Assume the following read history exists for a subtractive register (i.e., one where you have to subtract the previous read from the current read in order to derive consumption) where the register's unit of measure doesn't measure a peak quantity.

Read Date	Reading	Derived Consumption
15-Jan-1999	1000	N/A (first read)
15-Feb-1999	3000	2000
15-Mar-1999	4500	1500

Next, assume the meter's trend profile looks as follows:

Read Date	Total Qty (Assume kWh)	No of Units (Total Days)	Number of Reads	Average Consumption (Per Day)
13-Mar-1999	6,000,000	135,000	4,500	44.444444
14-Mar-1999	900,000	15,000	500	60
15-Mar-1999	5,000,000	137,750	4,750	36.297641
...				

13-Apr-1999	4,000,000	135,000	4,500	29.629630
14-Apr-1999	4,650,000	155,000	5,000	30

NOTE:

The system keeps a separate trend for every combination of trend area, trend class, unit of measure and time of use code. Trend area comes from the premise at which a meter is installed. Trend class comes from the SP type of the service point at which a meter is installed. Unit of measure and time of use come from the register read's register.

Next, assume a new register read is recorded on 15-Apr-1999 with a value of 5000.

In order to calculate estimated consumption for this register read, we have to calculate average daily consumption using the following formula:

$$\frac{\text{This customer's usage in previous period}}{\text{Avg customer's usage in previous period}} \times \text{Avg customer's usage in current period}$$

The following points describe exactly how the system calculates each variable in this formula:

- Determine **Average Customer's Usage In Current Period:**
 - Extract the number of reads from the meter's service point's SP type's trend class. We'll assume this is set to 7,500 reads.
 - Read trend records in reverse chronological order from the read date (assuming there is no trend record on 15-Apr-1999, we'll start amassing trends from 14-Apr-1999).
 - Read enough trend records until the number of reads is greater than or equal to the trend class' number of reads. We will have to read 2 trends records to amass this value (the ones on 14-Apr-1999 and 13-Apr-1999). We remember how many reads we extracted from the trend table for use a little later. In this case, we used 9,500 reads.
 - Divide **Total Qty** by **No of Units**. Using our example, we'd divide 8,650,000 kWh by 290,000 days to get 29.827586 kWh per day.
- Determine **This Customer's Usage In Previous, Non-Estimated Period:**
- Find the consumption associated with the previous, non-estimated read for the register's unit of measure / time of use codes at the service point. In this case, we'd find the read on 15-Mar-1999. Because this register is subtractive, we'd also need to find the previous, previous non-estimated read in order to calculate consumption. In this case, we'd find the read on 15-Feb-1999.

NOTE:

Minimum Days Between Readings. The algorithm parameter Minimum Days Between Readings controls the minimum number of days needed between the previous non-estimated read and the "previous, previous" non-estimated read.

- Divide the total consumption by the number of days. Using our example, we'd divide 1,500 kWh by 28 days to get 53.571429 kWh per day.

- Note: if the customer doesn't have consumption in the previous, non-estimated period, e.g., if it's a meter at a new premise, the system assumes the customer uses the same as the average customer's usage in the previous period (see next point).
- Determine **Average Customer's Usage In Previous Period**:
- Read trend records in reverse chronological order from the read date of the previous reading (15-Mar-1999).
- Read enough trend records until the number of reads is greater than or equal to the number of reads amassed when determining the Average Customer's Usage In The Current Period (the first point). Using our example, we'd have to read 3 trends records (the ones on 15-Mar-1999, 14-Mar-1999 and 13-Mar-1999).
- Divide **Total Qty** by **No of Units**. Using our example, we'd divide 11,900,000 kWh by 287,750 days to get 41.355343 kWh per day.
- Next, determine the number of days of estimated consumption. This will be equal the number of days between the estimation date and the prior reading (note, the prior reading could have been estimated). Using our example, we'd have 31 days (the number of days between 15-Apr-1999 and 15-Mar-1999).
- At this point, we have everything we need to estimate consumption. This will equal $((53.571429 / 41.355343) * 29.827586) * 31$ days. This is equal to 1,198 kWh.
- If we need to calculate high and low boundaries, we multiply 1,198 kWh by the high and low values defined for the register's unit of measure, the service point's service type and the read's read type.

Consumptive Register Where UOM Measures A Peak Quantity

The algorithm used to estimate consumption for a consumptive, peak unit of measure is identical to [Subtractive Register Where UOM Measures A Peak Quantity](#) with the exception that we don't have to find the previous, previous read in order to determine the customer's usage in the previous period.

Subtractive Register Where UOM Measures A Peak Quantity

The algorithm used to estimate consumption for a subtractive, peak unit of measure is identical to [Subtractive Register Where UOM Doesn't Measure Peak Quantity](#) the previous example except:

- The No of Units on the trend table is not the number of days. Rather, it's the number of reads that contributed to the trend.
- Because we don't care about number of days, we don't have to multiply final estimated consumption by the number of days in the estimation period.

Tips for Consumption Estimation Using Trends

The quality of the trend information and setup impacts your estimation results (depending on your estimation algorithm) and this in turn impacts any estimated bills you send your customers. When you are setting up consumption estimation parameters keep the following guidelines in mind.

Make Sure the Number of Reads on Trend Class Is Large Enough

You should make the Number of Reads on the [trend class](#) is large enough that individual reads do not cause statistical anomalies. For example, assume that a "normal" read in a trend class is 500 kWh and, that within the trend class, there is one "abnormal" customer using 500 percent of normal:

- If the number of reads in the class is 10, the average read is calculated to be 700 kWh. The abnormal read causes a 40 percent increase in the average.
- If the number of reads in the class is 100, the average read is calculated to be 520 kWh. The abnormal read causes a 4 percent increase in the average.
- If the number of reads in the class is 1000, the average read is calculated to be 502 kWh. The abnormal read causes a 0.4 percent increase in the average.

By sizing the number of reads in a trend class appropriately, you can reduce the impact of statistical anomalies.

NOTE:

Example Values. The example above is meant to illustrate the importance of using an appropriate number of reads in a trend class. The numbers used in the example should not be viewed as recommended values when setting up your trend classes.

Trend Classes and Trend Areas Should be Defined Appropriately

In addition to defining a large enough number of reads on the trend class, you want to make sure that the system can collect the number of reads in a relatively short period of time or the benefit of trends is diluted. For example, if the system collects the number or reads over a period of two days as opposed to two months, the seasonal trend information is better represented in your estimates.

Consider the following example. Assume the number of reads required in the trend sample is 1000 and your system collects the following trend information for the area/class:

Month	Number of Reads	Average
March	400	200 kWh
April	400	300 kWh
May	400	500 kWh

The trended average in May is 333.33 kWh $((80000 + 120000 + 200000) / 1200)$.

However, if the number of reads required in the trend sample is 1000 and your system collects the following trend information for the area/class:

Month	Number of Reads	Average
March	1100	200 kWh
April	1100	300 kWh
May	1100	500 kWh

The trended average in May is 500 kWh $(550000/1100)$, which is a more accurate representation of the trend in May.

NOTE:

Example Values. The examples above are meant to illustrate the importance of ensuring that the system collects enough reads in a short enough time span to accurately capture trends. The values are summarized by month and do not represent actual records in a trend profile.

The collection of an appropriate number of reads is a function of the number of reads defined for a trend class and the number of customers who are in each trend class/trend area combination. Make sure that the number of reads is not too large for the number of customers who are in each trend class/trend area combination and that the trend class/trend area combinations do not create groups of customers that are too small to calculate accurate trends.

Customers Should Be Classified Into Appropriate Trend Classes and Areas

Make sure that your customers are classified into appropriate trend area/trend class categories. If a customer's actual usage does not follow the trend (within a certain percentage) for the area/class in which the customer is classified, the consumption estimates for that customer may be inaccurate. For example, a residential customer should not be in a trend class with industrial customers.

A customer's premise references the trend area in which the customer is classified. A customer's service point type references the trend class in which the customer is classified.

Estimation and Negative Consumption

In some cases, a service point may be fitted with its own renewable energy source such as wind turbines or solar panels. The energy generated at this service point, as well as any energy used, could possibly be measured by a single meter, with the energy generated being netted against the energy used (a situation also referred to as net-metering). When the energy generated exceeds the energy used, negative consumption can result.

The possibility of negative consumption occurring at a particular SP/meter combination is indicated by the presence of a characteristic on the SP that identifies it as one that has generation capacity, as well as a characteristic on the meter type that identifies the meter as one capable of rolling backwards. Both characteristics must be present before negative consumption is allowed for that SP/meter combination. These characteristic types and values are defined on the Meter Management Options [Feature Configuration](#). The following points describe the various **Option Types** that must be defined if negative consumption is allowed by your implementation:

- Negative Consumption SP Characteristic Type. This option indicates the characteristic type used on service points to denote generation capacity.
- Negative Consumption SP Characteristic Value. This option indicates the characteristic value used on service points to denote generation capacity.
- Negative Consumption Meter Type Characteristic Type. This option indicates the characteristic type used on meter types to identify meters capable of rolling backwards.
- Negative Consumption Meter Type Characteristic Value. This option indicates the characteristic value used on meter types to identify meters capable of rolling backwards.

Because the factors that cause a meter to roll forward are very different from the factors that cause it to roll backward, it is impracticable for the estimation algorithms to try to derive an estimate in situations where negative consumption is allowed. The estimation rules simply cannot combine the effects of all these disparate factors into one meaningful figure. By extension, the same issue applies to the determination of high/low factors. Besides the fact that high/low validation is impossible without first obtaining an estimate, a meaningful set of high/low factors cannot be determined when it is not known how energy generated at the service point may have affected the consumption in the first place. Therefore, no estimation is done by the system for SP/meter combinations that allow negative consumption. Reads that belong to these SP/meter combinations are also excluded from the trending process so as not to skew the rest of the estimates for a certain trend area and trend class.

Setting Up Trend Areas

When you set up a premise, you must define the consumption trend area in which it is located. This categorization matters when consumption trends differ across your service territory AND you want the system to estimate consumption in different areas differently.

This categorization does not have to be done in respect of classic geographic boundaries like cities and counties. Rather, trend areas may be based on economic factors, climatic conditions, or anything else related to geography that affects consumption.

NOTE:

A premise's trend area will default based on its postal code. See [Setting Up Premise & Service Point Postal Defaults](#) for more information.

To define trend areas, open **Admin > Consumption > Trend Area**.

Description of Page

Enter a unique **Trend Area** and **Description** for every trend area.

Define the **Meter Read Estimate Algorithm** that is used to estimate consumption for meters installed at premises associated with this trend area. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that performs meter read estimation. Click [here](#) to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TREND_AREA](#).

Setting Up Trend Classes

When you set up an SP type, you must define the consumption estimation trend class in which its consumption will be categorized. This categorization matters when consumption differs based on the type of property AND you want the system to estimate consumption in different classes differently.

These categories could be the classic divisions of residential versus commercial versus industrial consumers. Alternatively, they could be finer-grained divisions: single family residence, versus duplex, versus triplex, versus medical office, versus grocery store.

To define trend classes, open **Admin > Consumption > Trend Class**.

Description of Page

Enter a unique **Trend Class** and a **Description** for every trend class.

Enter the **Number of Reads** that must be amalgamated to create a statistically significant sample when amassing the average customer's consumption. The system uses this number to determine the number of consumption trend records to amalgamate.

NOTE:

Don't worry. The Number of Reads is not the number of rows that will be read when the system estimates consumption. This is because the total consumption from many reads is stored on a single consumption trend record.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TREND_CL](#).

Setting Up High / Low Factors

When consumption estimation is done for the purpose of meter read validity, the system checks if the meter read's consumption is reasonable. The High / Low Meter Read algorithm type ([HILO-FACTBL](#)) supplied with the base package uses the factors defined in this table to calculate the high / low values.

High and low factors are defined for every measurable unit of measure (UOM) and read type combination. The word measurable was underlined because some UOMs exist purely for pricing purposes and are never measured by a meter (e.g., a therm of gas).

FASTPATH:

For more information, refer to [High / Low Checks](#).

To define high / low factors, open **Admin > Device > High Low Factor**.

Description of Page

Enter the **Unit of Measure** (UOM) for which the high / low factors are used.

Enter the **Read Type** for which the high / low factors are used. Valid values are: Billing Force, Customer Read, Office Estimate, No Read , Service Provider Estimate, System Estimate, Regular, and Verified.

Enter the **Low Factor** for the UOM and Read Type. This value will be multiplied by estimated consumption to derive the acceptable low value of a meter read with this UOM and read type.

Enter the **High Factor** for the UOM and Read Type. This value will be multiplied by estimated consumption to derive the acceptable high value of a meter read with this UOM and read type.

Where Used

This information is used by the High / Low Meter Read algorithm type ([HILO-FACTBL](#)) when the system calculates the low and high consumption amounts against which a meter read's consumption is compared when the read is added to the system.

NOTE:

Detecting Theft Of Service. When the status of a meter is Off, the system sets the high and low read values equal to the previous register read. This catches any consumption activity at the service point after a meter has been turned off. However, for some implementations, a small amount of consumption trickle at the service point is acceptable. In this case, you would want to relax the high/low thresholds so that the reads are not constantly flagged to have failed high/low validation. To do this, you can add the Always Estimate And Apply High/Low Factors **Option Type** under Meter Management Options [Feature Configuration](#) and set its value to Y .

Setting Up Trends

FASTPATH:

Refer to [Setting Up Trends](#) for more information.

Setting Up Meter Read Options

This section describes tables that must be set up before you can enter a meter read.

Setting Up Meter Reader Remarks

When you enter a meter read, you may define remarks using remark codes. The topics in this section describe how to set up meter reader remark codes.

Meter Reader Remark - Main

To define meter read remark codes, open **Admin > Device > Meter Reader Remark > Add**.

Description of Page

Enter a unique **Meter Reader Remark** and a **Description** for every meter read remark.

Turn on **Eligible for Processing** if meter reads marked with a given remark code should cause one or more **Action Algorithms** to execute. For example, if you need a "reread" field activity to be created when a "high bill" remark is uploaded, you'd create an algorithm called "reread" and associate it with the "high bill" remark code. Then, whenever such a meter read with a "high bill" remark is recorded in the database, the system will execute the algorithm (and generate the field activity). If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that is associated with a meter read remark special activity. Click [here](#) to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_READER_REM](#).

Meter Reader Remark - Bill Messages

To define bill messages to appear on bills that use a meter read that references a given meter reader remark code, open **Admin > Device > Meter Reader Remark > Search** and navigate to the **Bill Messages** tab.

Description of Page

Use the **Bill Messages** collection to define **Bill Message** codes that should appear on bills that use a meter read that references a given meter reader remark code. For each message, also specify the **Start Date** and **End Date** when such a message should appear on the bill (leave **End Date** blank if the message should appear indefinitely).

Where Used

The system snaps bill messages on a bill during bill completion. Refer to [The Source Of Bill Messages](#) for more information.

Setting Up Meter Read Sources

When you add or upload a meter read, you may define the source of the meter read. The source could reference a specific reader, a meter reading agency, or any other possible source. To define meter read sources, open **Admin > Meter Read Source**.

Description of Page

Enter a **Meter Read Source** code and **Description** for every meter read source.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MR_SOURCE](#).

Setting Up Items

This section describes tables that must be set up before you can define items.

Setting Up Item Types

Every item has an item type that defines characteristics common to all items with this type. The topics in this section describe how to set up your item types.

Item Type - Main

To define item types, open **Admin > Item Type > Add**.

NOTE:

When a new item type is added. After adding a new item type, you must define the SP types at which items of this type can be installed.

FASTPATH:

Refer to [Service Points \(SPs\)](#) for information about the difference between badged and non-badged items. Refer to [Items Are Used For Other Devices Associated With A Customer's Service](#) for more information about items in general.

Description of Page

Enter an easily recognizable **Item Type** and **Description** for the item type.

Choose the **Service Type** associated with all items of this type. Refer to [Setting Up Service Types](#) for more information.

Turn on **Use Estimates** if estimated consumption is recorded for this type of item (e.g., lamps have estimated consumption that is used to calculate billable charges for these types of items). When this switch is turned on, also define the **Unit Of Measure** in which the estimated consumption is expressed.

FASTPATH:

For more information about estimated consumption, refer to [Setting Up Estimated Consumption For Items](#).

Turn on **Summarize For Billing** if billing is supposed to summarize all items of this type on a customer's bill. If billing is supposed to show a separate bill line for every individual item of this type, turn this switch off.

WARNING:

The Summarize For Billing switch is only pertinent if badged items are linked to the service points being billed. If unbadged items are linked to the service points, the bill will contain a summary of items by item type regardless of the value of this switch. Refer to [Metered versus Item-Based versus Non-Badged Service Points](#) for more information.

Turn on **Multiple Equipment Assignment** if equipment of this type can be linked with more than one service point / meter / item at any instant in time.

Turn on **Billable** if billing should amalgamate items of this type when it calculates a bill segment. This switch would typically only be turned-off for items used to describe [equipment](#) and other non-billable items linked to a service point.

WARNING:

If this switch is turned on, rates linked to service agreements used to bill for items of this type must include an [Item Type Calculation Rule](#) that references this item type. Why? Because we assume that a bill line should be produced for "billable" items (and bill lines are produced for items using "item type" calculation rules). If you neglect to have such a calculation rule, a bill segment error will be produced.

If your organization bills for items of this type using [Estimated Consumption](#) rather than on a "per item" basis, you'll still need an "item type" calculation rule in these rates (otherwise billing will generate an error). In order to prevent the system from calculating a charge for such an "item type", make sure to indicate the calculation rule is "for calculation purposes only" (FCPO). Refer to [Common Parameters](#) for Calculation Rules for more information about FCPO's. Also note that you might also want to indicate that the calculation rule is not printable if you want to suppress the item type's description on the printed bill. Refer to [How to Use Description on Bill](#) for more information about printable calculation rules.

Indicate in **Track Location** whether you Track or Do Not Track the location of items of this type. Refer to [The Big Picture of Asset Inventory](#) for more information.

Use the **Characteristic Values** collection to define **Characteristic Types** and their respective **Characteristic Values** to describe characteristics common to all items of this type.

NOTE:

You can only choose characteristic types defined as permissible on the item type record.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_ITEM_TYPE](#).

Item Type - Item Characteristics

To define characteristics that may be defined for items of a given type, open **Admin > Item Type > Search** and navigate to the **Item Characteristics** page.

Description of Page

Use the **Item Characteristics** collection to define characteristics that can be defined for items of a given type. Turn on the **Required** switch if the **Characteristic Type** must be defined on items of a given type. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

Item Type - Equipment Types

Open **Admin > Device > Item Type > Search** and navigate to the **Equipment Types** page to define the types of equipment that can be linked to items of a given type. Refer to [Equipment versus Badged Items](#) for more information about equipment.

Description of Page

Use the collection to define the item types of **Equipment** that can be linked to items of this type.

NOTE:

Item types are being specified. There is no equipment type control table. Items are used to define equipment; therefore, you define item types instead of equipment types. Refer to [Equipment versus Badged Items](#) for more information about equipment.

Item Type - Test Types

Open **Admin > Device > Item Type > Search** and navigate to the **Test Types** page to define the types of device tests that can be performed on items of a given type. Refer to [The Big Picture Of Device Testing](#) for more information about device tests.

Description of Page

Use the collection to define the types of **Device Tests** that can be performed on items of a given type.

Setting Up Manufacturers and Models

When you set up an item (or a meter) you must define the manufacturer and model number of the meter / item.

FASTPATH:

Refer to [Setting Up Manufacturers & Their Models](#) for more information.

The Big Picture Of Device Testing

The topics in this section describe meter and item (i.e., device) testing at a high level.

We strongly recommend examining the demo data to see how the concepts explained in this section could be used in real life.

WARNING:

Setting up the device testing control tables is as challenging as your organization's business rules. If you don't test your devices, you don't have to setup anything. If you have sophisticated testing requirements, your setup process will be more challenging.

The Level of Complexity Depends On What You Test and Your Record Keeping Requirements

Every organization's device testing requirements are different. Consider the following:

- Some organizations simply use the CIS system to create and dispatch field activities when it's time to test devices. After the field activities are dispatched, the test results are maintained in a separate system. You can set up the system to work this way for you.
- Some organizations maintain very detailed test result records, others don't. For example, some organizations record both "as found" and "as left" test results, i.e., they test the meter in its "as found" state, then they recalibrate it and test it again and record the "as left" state. You can set up the system to work this way for you.
- Some organizations maintain separate test results for every individual register on a meter. For example, they might record 3 separate test results for an electric meter with kWh, kW, and kVar registers (and, if they recorded both "as

found" and "as left" results, they would have 6 separate test results, at least, for this meter). You can set up the system to work this way for you.

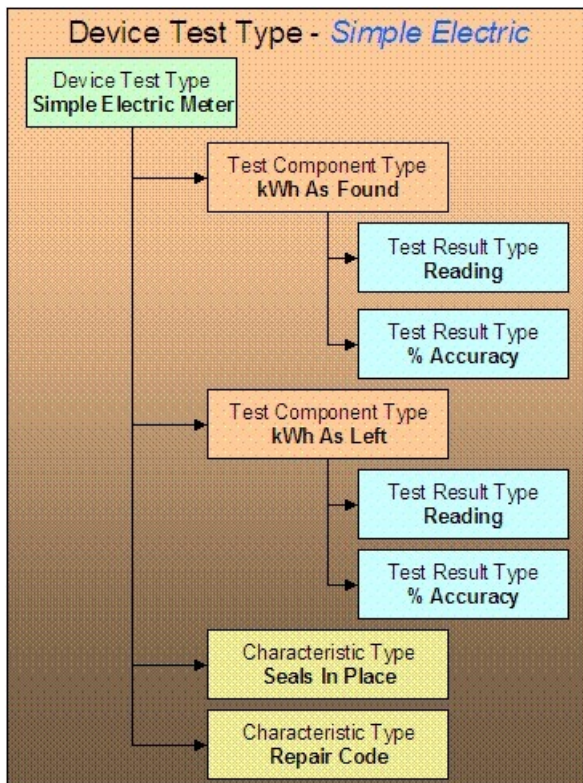
- The type of information that can be recorded in respect of a test varies widely. Some organizations simply record if the test passed or failed. Other organizations keep track of a great deal of information about the test results. Consider the following examples:
 - Most organizations record who performed the test, the state of the seals on the meter, and whether or not a repair is necessary.
 - A company with gas meters might record the ambient air temperature, the temperature of the oil used in the test, the test spin time, the actual meter reading, the "chart" (should be) reading and the percent accuracy.
 - A company with electric meters might record the following information: meter reading at full capacity, meter reading at a light capacity, meter reading with a power factor of 50%, and the percent accuracy.
- You can set up the system to work this way for you.

Given that the system supports the above disparate requirements, you can understand why the set up process is either straightforward or challenging. The remaining topics in this section provide some guidelines to help you through this setup process.

A Device Test Records Test Results

You create a device test each time you test a meter or item. The device test keeps track of when the test was conducted, who conducted it, and the results of the test. Every device test references a *device test type*.

When you design your device test types, you are actually defining the type of information that can / must be recorded when such a device test is saved in the system. The following picture illustrates a device test type used for tests of simple electric meters.



Notice that the above device test type example uses two different mechanisms to record test results:

- **Component tests.** If your organization maintains the test results from individual registers, you will use component tests. In the above example, two different types of component tests can be recorded for this type of device test: one is used to record a register's accuracy before calibration (the "as found" component test), another is used to record the register's accuracy after calibration (the "as left" component test).
- The component test type controls the type of information that is recorded for a component test. In the above example, each component test type requires the same result types - a register reading and a percent accuracy. Note, the fact that these two test types require the same results is coincidental.

You have to set up a component test types and test result types to satisfy your organization record keeping requirements.

- **Characteristics.** If your organization doesn't keep register-specific test results, you don't have to use component tests. Rather, you can simply use characteristics to record test results. In the above device test type, characteristics are used to record whether the seals were in placed and a repair code (if the test results in a repair). Refer to [Setting Up Characteristic Types & Their Values](#) for more information about characteristics.

Field Activities And Device Testing

Most device tests occur as a result of a field activity. The field activity is associated with the service point at which the device is currently installed. Refer to [Examples of Device Testing Activity Types and their Steps](#) for an overview of how these types of field activities look.

You can create device testing field activities manually OR you can take advantage of the [Device Test Selection](#) page. This page will generate field activities to test meters and items based on a user-defined Test Selection Algorithm. The system comes with a sample Test Selection Algorithm type that selects meters of a given manufacturer / model that haven't been test for a given number of months. In all likelihood, you or your implementer will have to develop other algorithms to meet the test selection requirements of your organization. For example, if you test all meters that generate more than \$100,000 of revenue per annum every 6 months, you will have to write a new Test Selection Algorithm Type.

After defining which test selection algorithm types you need, you must set up Test Selection Algorithms that make use of them (these algorithms are specified by the operator on the [Device Test Selection](#) page). To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that is associated with a device test selection. Click [here](#) to see the algorithm types available for this plug-in spot.

Device Test Validation

After entering the test results for a device test, some organizations want the system to analyze the test results and determine if the meter / item passed or failed the test. If your organization does this, you will have to determine the characteristic values and test result values associated with "passing the test". You will then have to design an algorithm that contains logic that determines if a device test contains the appropriate characteristic and test result values.

After developing your device test algorithms, you associate them with the appropriate device test types. The system invokes the respective algorithm when an operator pushes the Validate Device Test button on the device test page.

Setting Up Device Test Options

The topics in this section describe how to set up the control tables that must exist before a device test can be recorded.

Setting Up Component Test Types

If your organization records component test results, you must set up a component test type for each type of component test. Refer to [A Device Test Records Test Results](#) for more information about component test types.

Open **Admin > Device > Device Test Component Type > Add** to define the type of information that must be recorded on a component test of a given type.

NOTE:

When a new component test type is added. After adding a new component test type, you must define the device test types that make use of it.

Description of Page

Enter an easily recognizable **Test Component Type** and **Description**.

Turn on **Meter Test** if this type of component test is associated with a meter. If this type of component test is associated with an item, this switch should be off.

If **Meter Test** is on, turn on **Register Required** if a register must be referenced on component tests of this type. You would require a register if a) you keep component tests for individual registers, or b) if your component tests require the entry of register readings (as defined in the following grid).

The grid that follows defines the type of test results that are recorded.

Result Seq Result sequence controls the order in which the test results are captured.

Description Enter the prompt that is displayed for this type of test result.

Result Required Turn this switch on if a result must be specified on a test component of this type.

Validation Algorithm The validation algorithm controls how the test result is validated. Leave this field blank if no validation should be imposed.

Prompt For Read Turn on this switch if a user can enter a register read for this type of test result.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TST_COMP_TYP](#).

Setting Up Device Test Types

Every device test references a device test type. The device test type controls business rules associated with its device tests. The topics in this section describe how to set up your device test types.

Device Test Type - Main

Open **Admin > Device > Device Test Type > Add** to define general business rules shared by device tests of a given type.

NOTE:

When a new device test type is added. After defining your device test types, you must update your meter and item types to define their valid test types. This is what prevents a gas test from being performed on an electric meter.

Description of Page

Enter an easily recognizable **Test Type** and **Description**.

Turn on **Meter Test** if this type of test is associated with a meter. If this type of test is associated with an item, this switch should be off.

Turn on **Validation Required** if tests of this type use a **Validation Algorithm** to determine if the test results (and therefore the device test) are of a passing grade. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that is associated with a device test validation algorithm. Click [here](#) to see the algorithm types available for this plug-in spot.

Define **Characteristic Types** and their respective **Characteristic Values** to describe characteristics common to all tests of this type. Note that you can also define characteristic types for which values are required to be entered when the device test is created (see [Device Test Type - Characteristics](#)).

NOTE:

You can only choose characteristic types defined as permissible on a device test type record.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_DV_TEST_TYPE](#).

Device Test Type - Component Types

Open **Admin > Device > Device Test Type > Search** and navigate to the **Component Types** page to define the types of component tests (if any) that can be performed for device tests of a given type.

Description of Page

Use **Sequence** to define the relative order of each **Test Component Type** that can be performed during device tests of a given type.

Device Test Type - Characteristics

Open **Admin > Device > Device Test Type > Search** and navigate to the **Characteristics** page to define required and optional characteristics for device tests of a given type.

Description of Page

Define **Characteristic Types** to describe characteristics that might be entered for all tests of this type. Turn on **Required** if this type of characteristics must be specified on tests of this type, otherwise, the characteristic will be optional. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

NOTE:

You can only choose characteristic types defined as permissible on a device test type record.

Usage Administration

This section describes concepts and common tasks related to usage administration.

The Big Picture Of Usage Requests

Some organizations use a meter data management (MDM) system to record meter reading information. Since meter reads are not available in Oracle Utilities Customer Care and Billing to calculate consumption, during the billing process usage (or bill determinants) must be requested from the MDM system.

The following sections describe how the base-package usage request process works.

Requesting Bill Determinants

The term bill determinant request is another way to refer to a usage request. These usage requests are created during the batch billing process, and also when a user generates an online bill segment or performs cancel / rebill. The get consumption algorithm specified on the SA Type's Bill Segment Type is responsible for creating the usage request.

Batch Billing Usage Requests

Batch billing usage requests are created and held in the initial pending state. A separate batch process is responsible for transitioning these requests and sending them to the MDM system. Here's how this works:

- The get consumption algorithm specified on the SA Type's Bill Segment Type is responsible for creating the usage request. These batch billing usage requests are held in the Pending state until the batch Usage Scheduled Monitor Process is executed
- Similarly, once a response is received from MDM, the usage request will not transition to the Bill Determinants Processed state until the batch Usage Scheduled Monitor Process is executed
- In addition to the standard MDM usage request elements, the system captures the bill cycle and window start date.

Online Billing Usage Requests

- Usage requests created from an online billing request are not held in the initial state. These usage requests are transitioned through their lifecycle resulting in the request being sent to MDM and, once a response is received, the bill segment is regenerated.
- The bill segment remains in the Freezable state until the user freezes the bill segment and completes the bill. If however, the freezable bill segment is regenerated, the system cancels the usage request and creates another usage request for the bill segment.
- In addition to the standard MDM usage request elements, the system captures the responsible user id.

NOTE:

External Reference ID. When creating a usage request, the external reference id is populated based on the version of MDM that your implementation integrates with. This is defined as the MDM Version MO option on the Usage maintenance object. If integrating with MDM 1, the system populates the external reference id using the batch run thread scheduler id to facilitate processing in the external system. Otherwise, the external reference id is left blank and may be updated on the response from the external system.

Usage Request Lifecycle

The usage request business object (BO) contains the rules that govern the processing of a usage request. The base product provides the BO C1-UsageRequest which serves as a parent BO and contains the following lifecycle:

- All usage requests are created in the initial Pending state. If the usage request was created from the batch billing process, the batch Usage Scheduled Monitor Process must be executed to transition the usage request.
- If there are pending sync requests for the usage request's service agreement, the usage request is held in the Awaiting Data Sync state until the sync is processed.
- Once all pending sync requests related to the usage request have been processed, the usage request transitions to the Send Request state. An enter algorithm on this transitory state is responsible for sending the usage request to MDM. The base algorithm creates a JMS Queue outbound message. This outbound message is configured to not persist on the database.
- Once the outbound message has been successfully sent, the usage request sits in the Awaiting Bill Determinants state until a response is received from MDM, or a time out is encountered.
- When a response is received from the external system, the usage request is transitioned to either the Bill Determinants Received state, or the Error state. In the case of an error, the following occurs:
 - If the usage request was created from the batch billing process and the billing window is still open, a new usage request will be created the next time billing runs and the process is repeated. If however, the billing window is no longer open, a to do entry is created for manual follow up.
 - If the usage request was created from an online billing process, the user will be notified. The bill segment remains in the error state and the user can either delete the bill segment, or attempt to generate it again. If the bill segment is regenerated, a new usage request is created and the process is repeated.
 - The system transitions usage requests in the Bill Determinants Received state to the Bill Determinants Processed state. An enter algorithm on this final state is responsible for generating the bill segment. If the usage request was created from the batch billing process, the bill segment is frozen and the bill is completed. For online billing usage requests, the user is responsible for freezing the bill segment and completing the bill.

Summary Billed Accounts and Usage Requests

A "summary" account refers to an account with service agreements that cover more than one premise. If these premises are linked to service points that span different service cycle routes scheduled for reading on different dates, there's no guarantee that all the account's SAs will be billed at the same time. This is because batch billing for manually read meters is based on the meter's scheduled reading date.

For example, assume a summary account has two SAs each linked to a different premise. One is on route X, scheduled for reading every first of the month and the other is on route Y, scheduled for reading every 20th of the month. Assume that the account's bill cycle is cycle X (which matches the schedule of premise/SP on route X). When billing runs on the first of the month, the service agreements tied to route Y won't have any reads. And because route Y is not scheduled for reading, the system will not estimate the consumption. Instead, it will skip billing the route Y SAs until the next month. This means some SAs are always one billing cycle behind.

If your implementation maintains meters and meter reads in CCB, then the base get consumption algorithm called *Get Consumption From SP's Linked To SA* handles the skipping of a summary billed account's service agreements based on bill cycle and service cycle schedules. The remainder of this section describes how this works in a CCB-MDM integrated environment.

Requesting Bill Determinants for Summary Billed Accounts

Usage requests for summary billed accounts are handled as follows:

- Assuming MDM owns meter read schedules; when batch billing executes CCB cannot determine whether a summary account's service agreement should be included on the bill or not.
- On the first night of the billing window usage requests are created for each SA and sent to MDM. MDM then checks what the next schedule read date is for each service point.
- If MDM determines that an SA/usage subscription should be skipped from billing, it notifies CCB by including a skip indicator and reason on the usage response, as well as the SA's next scheduled read date. This date is captured as a **MDM Next Scheduled Read Date** characteristic on the SA, and the usage request is cancelled. The system still attempts to complete the bill since this might be the last outstanding usage request for the bill
- When billing next runs, the base *Get bill segment consumption using a usage request* algorithm first checks if the SA should be skipped as follows (Note that this check is also performed in first billing run):
- Get the **MDM Next Scheduled Read Date** characteristic on the SA. If this date is after the bill segment end date, we'll skip the SA from billing

If your implementation would like to wait until the bill cycle window end date before attempting to complete summary account's bills, ensure that the **Complete Summary Bill at End of Bill Cycle** pre-bill completion algorithm is defined on customer class. This is useful if scheduled reads are expected in the bill cycle window, but not necessarily on the window start date.

Corrected Read Notifications

If a read that was used for bill determinant calculations is modified in the MDM system, notification is sent to Oracle Utilities Customer Care and Billing. This results in the creation of an off cycle bill generator. The system uses the business object defined as the OCBG Corrected Read BO MO option on the Off Cycle Bill Generator maintenance object to create the OCBG. If any errors are encountered while attempting to create the OCBG, the system sends a message to the external system using the Outbound Message Type and External System defined as BO options on the OCBG Corrected Read BO

What happens next depends on the lifecycle that your implementation has configured for the OCBG Corrected Read BO. Here are examples of what might occur:

- Create a to do entry for manual follow up.
- Find any frozen bill segments that might be affected by the corrected read and perform cancel / rebill.

Configuring The System For Usage Request Integration

Oracle Utilities Customer Care and Billing sends usage requests to MDM in the form of an xml message. These messages are transformed by the integration layer and then sent to MDM. Similarly, MDM sends responses to the integration layer so that the data can be transformed and sent to Oracle Utilities Customer Care and Billing.

The following sections describe at a high level the data setup required to send usage requests to an MDM system.

Define the Outbound Message Type

An outbound message type is required for the batch billing usage request outbound message. This outbound message type must reference the base C1-CyclicalUsgReqOutMsg business object. The outbound message type must also be specified as a BO option on the base C1-CyclicalUsgReqOutMsg business object so the system knows which outbound message type to use when sending usage requests to MDM.

An outbound message type is required for the online billing usage request outbound message. This outbound message type must reference the base C1-NonCyclicalUsgReqOutMsg business object. The outbound message type must also be specified

as a BO option on the base C1-NonCyclicalUsgReqOutMsg business object so the system knows which outbound message type to use when sending usage requests to MDM.

NOTE:

Defining the Usage Business Objects. The business objects used by the get consumption algorithm when creating a batch or online billing usage request are defined as MO options on the Usage maintenance object.

Define the Message Sender

A Message Sender is required to define how to send usage requests to MDM. Use the context of the Message Sender to define the web service interface.

Define the External System and Configure the Messages

Define an external system and configure the valid outbound message types and the method of communication for each. You will also need to select the appropriate XSLs to format the request for usage. The external system must also be specified as a BO option on the base C1-CyclicalUsgReqOutMsg and C1-NonCyclicalUsgReqOutMsg business objects so the system knows which external system to use when sending usage requests to MDM.

FASTPATH:

Refer to the *Oracle Utilities Customer Care and Billing - Meter Data Management Integration Implementation Guide* for more information.

Designing Your SA Types For Usage Requests

SA Types used to bill service agreements that require bill determinants from a meter data management system must have the following characteristics:

- The SA Type must have a special role flag of Bill Determinants Required
- Bill segment type:
 - Reference the bill segment creation algorithm that creates a bill segment from a usage request.
 - Reference the bill segment get consumption algorithm that gets bill segment consumption using a usage request.

Start And End Times For Billing

As you know, there is logic in billing to determine the start date and end date for a bill segment. Refer to [Ways to Control The End Date Of A Bill](#) for more information. When billing for a customer with interval data, the system also needs to know the time.

The time used by billing, referred to as the cutoff time, is stored on the service agreement. There is also a control on the service agreement called Start Day Option that determines which day to use for the start time. Billing uses the billing date, the cutoff time, and the start day option to determine the correct interval data to process.

When integrating with a meter data management system, Oracle Utilities Customer Care and Billing lacks knowledge of the type of meter installed at a service point. Thus, both interval processing period as well as scalar processing information is captured on a usage request. MDM then uses the appropriate period to calculate bill determinants based on the type of meter installed.

MDM returns the true usage period used to calculate bill determinants on the usage response. This usage period is captured on the bill segment.

FASTPATH:

Refer to [Start and End Times for Billing](#) for more information.

Navigating To MDM

It might be necessary for Oracle Utilities Customer Care and Billing users to navigate to MDM to view detailed read information for a service point. For example, a user may want to see the unbilled consumption that's available in MDM. The following is required to implement this:

- Add a menu entry to the service point context menu. This menu item must reference the base Go To MDM From SP navigation option
- Add a menu entry to the service agreement context menu. This menu item must reference the base Go To MDM From SA navigation option
- Set up the MDM URL option type on the General System Configuration [Feature Configuration](#). The MDM URL option must contain the URL for the MDM application.

Once configured, users should be able to launch the MDM system from these context menus with either a service point or service agreement id in context.

The Big Picture of Time of Use Mapping and Pricing

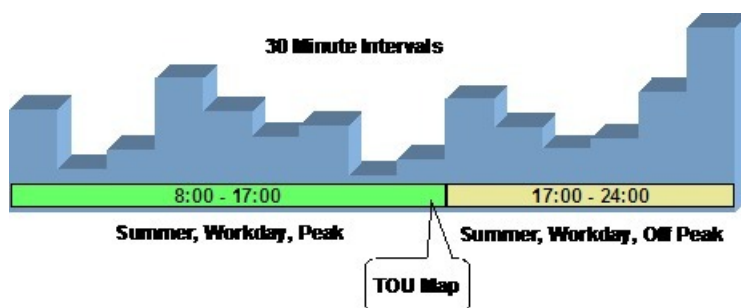
This section provides an overview of the concepts related to setting up your control tables to support time of use mapping and pricing with the rates.

NOTE: Time of use mapping is supported to enable low volume interval data manipulation (for example, for mapping electric vehicle charging events into time of use quantities). A meter data management application, such as Oracle Utilities Meter Data Management, must be used to manipulate high volumes of interval data and calculating bill determinants for billing purposes.

Mapping to Time of Use (TOU) Periods

Many customers choose not to price their interval data using interval prices. Customers may choose for their interval data to be mapped into time of use (TOU) periods. This option for interval data might be preferred because:

- Typically it involves fixed prices for the use periods
- It is more manageable than direct billing
- It is easier for a customer to forecast and budget



A TOU map's purpose is to define the TOU codes for a collection of time period definitions (i.e. given dates and times). The TOU Map has a TOU Map Type, which defines the interval size between TOU map data rows.

Time of use periods can (and often do) change during the year.

Map #123 (TOU Group 2)		
Effective 1 Jan 2000		
Interval Date/Time		
30/Apr/00	16:30	On Peak/Winter
30/Apr/00	16:45	On Peak/Winter
30/Apr/00	17:00	Off Peak/Winter
30/Apr/00	17:15	Off Peak/Winter

01/May/00	7:45	Off Peak/Summer
02/May/00	8:00	On Peak/Summer
02/May/00	8:15	On Peak/Summer

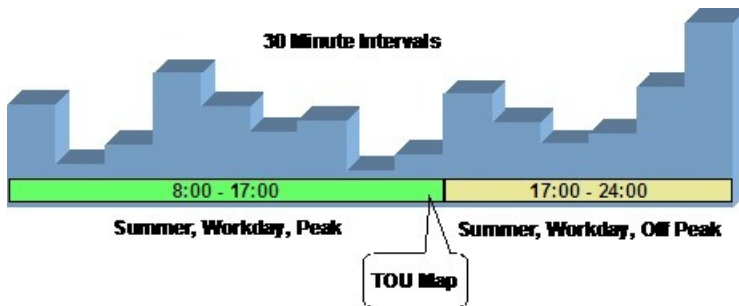
All the possible TOU codes for a given map are grouped together in a TOU group. Refer to [Grouping of TOU Codes for TOU Mapping](#) for more information.

Grouping of TOU Codes for TOU Mapping

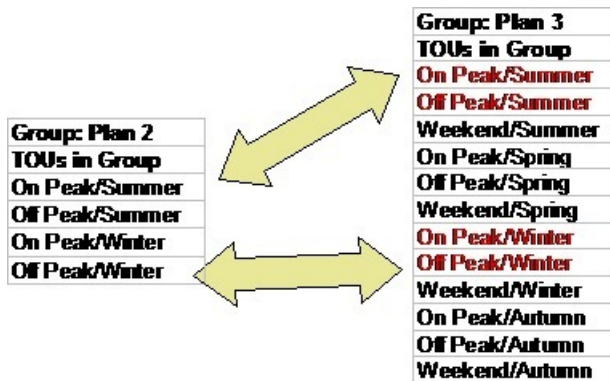
This section describes the relationship between TOU Codes and TOU Mapping.

Overview of TOU Codes and TOU Mapping

A TOU map's purpose is to define the TOU codes for a collection of time period definitions (i.e. given dates and times).



Time of use codes or TOU codes (for example, "Off Peak/Winter", "Off Peak/Summer") are user-defined. Refer to [Setting Up Time-Of-Use Codes](#) for more information.



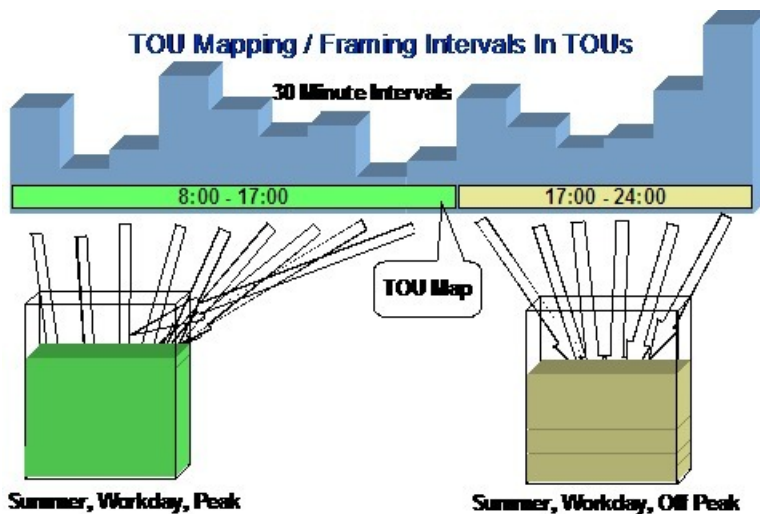
TOU Sequence for TOU Mapping and Pricing

If desired, you may use sequence number to indicate the relative position or relative priority of each TOU code within a TOU group. This sequence number is not used by any system functionality, but is available for you to use in a plug-in algorithm.

Time of Use (TOU) Mapping and Pricing

The following section describes the logic used by the system to map an interval data curve to TOU periods using a TOU map, and subsequently apply prices to these mapped quantities.

At some point during billing, the system will take an interval data curve and will map the interval values to TOU periods based on a TOU Map. The interval data curve is retrieved from either a specific usage request or from the new rate engine buffer populated prior to calling the rate engine.



This type of data manipulation is typically handled using a calculation rule based on the 'math' calculation rule type.

Once the interval data curve values have been mapped to TOU periods, prices can be applied to the time of use period quantities using calculation rules based on the 'service quantity' calculation rule type.

Calculation Rules for TOU Mapping & Pricing

Calculation Rules are used to perform time of use mapping and time of use pricing.

Time of Use Mapping

Time of use mapping will be typically handled using a calculation rule based on the 'math' calculation rule type. Calculation rules of this type can perform many functions. Two examples include:

- Applying a TOU map to an interval data curve to produce service quantities that will be added to the bill segment's SQ collection.
- Applying a TOU map to a derived interval data curve. For example, after deriving the power factor curve, perform TOU mapping on the resultant curve.

For further details on the 'math' calculation rule type, refer to [Base Package Math Calculation Rule](#) for more information.

Time of Use Pricing

Time of use pricing will be typically handled using calculation rules based on the 'service quantity' calculation rule type. Calculation rules of this type can perform many functions. For example:

- Add or update a service quantity in the bill segments SQ collection
- Create bill calculations lines that levy charges based on some type of consumption. For example, applying a price to a specific service quantity in the bill segment's SQ collection

For further details on the 'service quantity' calculation rule type, refer to [Base Package Service Quantity Calculation Rule](#) for further information.

NOTE: Calculation rules are implemented using algorithms that contain the logic to be performed. If your implementation requires a new type of calculation rule, you can configure a new calculation rule type and reference the appropriate algorithm on it.

TOU Map Used For Mapping

An interval data curve is mapped into time-of-use periods using a TOU map. The TOU map used by the calculation rule for mapping interval data curve values to time of use periods is defined directly on the calculation rule.

Attributes of TOU Maps

TOU map types define attributes shared by TOU maps of a given type. The essential attributes of any TOU Map Type are the interval sizes between TOU map data rows, associated TOU map template(s) to use, and whether the TOU data will follow any seasonal time shifting.

Designing Your Time of Use (TOU) Mapping and Pricing Options

This section provides an overview for designing your control tables to support time of use (TOU) mapping and pricing with the new rating engine.

Designing Your Time Of Use (TOU) Mapping and Pricing Calculation Rules

As you know from the rates chapter, the system can handle mapping of interval data curve values into different time of use periods and pricing these quantities using the 'math' and 'service quantity' calculation rule types respectively. Refer to [Base Package Calculation Rule Descriptions](#) for more information.

The following guidelines provides a high level outline on how you should go about designing a rate schedule that involves time of use mapping and pricing.

- Obtain copies of existing bills that use the rate in question. If the rate is new, then write up exactly how the information should appear on the customers' printed bills.
- Identify all the lines that represent charges for individual time of use periods.
- Determine how the quantities for the time of use periods are calculated. Which time of use map is used to define the time periods?
- In order to perform time of use mapping and pricing, calculation rules need to be designed
 - A calculation rule based on the 'math' calculation rule type may be used to map an interval data curve into time of use periods. This calculation rule must know the TOU map to apply to the interval data curve to map into the relevant time of use periods. The time of use period quantities are populated in the bill segments SQ collection.
 - Calculation rules based on the 'service quantity' calculation rule type may be used to apply prices to each mapped time of use period quantity. These calculation rules each require either a price or regular bill factor for pricing each time of

use period quantity. The bill factor may contain the price directly, or may indicate that the price is customer specific and can be found as contract quantities for the service agreement.

Once you have your calculation rules designed, you will be able to design the other control tables needed to set up your time of use billing customer.

Designing Your TOU Codes for TOU Mapping

The next most logical step in designing your time of use mapping controls is to define your time of use codes. To do this, look at the time of use periods to which your usage needs to be mapped. These values will likely correspond to the time of use quantities that your rate bills for. It should be noted that it's possible that will not bill for every time of use period.

For more information about time of use, refer to [UOM versus TOU versus SQL](#).

Designing Your TOU Groups for TOU Mapping

To further aid in designing time of use mapping, the TOU Group enables you to group together all the time of use codes that are available to be used in a single map. You must also decide if you want to use a sequence number to define the relative order of a TOU code within a TOU group.

When TOU data is created for a TOU map, only TOU periods defined on a specified TOU group can be specified.

Designing Your New Style TOU Map Types for TOU Mapping

Now that you have your TOU groups defined, you can begin defining TOU map types. Recall that the TOU map type defines the attributes that may be shared by TOU maps of the same type.

The essential attributes of any TOU Map Type are the interval sizes between TOU map data rows, associated TOU map template(s) to use, and whether the TOU data will follow any seasonal time shifting. Refer to [Designing Your Time Options](#) for more information.

TOU Map Type Interval Size

The interval size of a TOU map must divide evenly into the interval size of the interval data curve that uses the map (because the system joins the date/time of the interval data curve values to the date/time of the TOU data). This means that it is possible to use a 15 minute TOU map with a 60 minute interval data curve. However, it is not OK to have a 60 minute TOU map used with a 15 minute interval data curve because the join will miss 3 out of 4 interval data curve values.

TOU Map Type Override Template

While most TOU maps will use the TOU map template defined on the TOU map type, TOU maps also support override templates. Refer to [Designing Your New Style TOU Map Templates](#) for TOU Mapping for more information.

- A TOU map's TOU map type defines the default TOU map template that's used to generate its TOU data.
- A TOU map's type defines the TOU map templates that can be referenced on individual TOU maps to override the default template.
- An individual TOU map can have override templates. If the TOU map doesn't have an override template, the default template defined on the TOU map type is used to generate the map's TOU data.

Refer to [Creating New Style TOU Map Types](#) for further information.

Designing Your New Style TOU Map Templates for TOU Mapping

In order to help your users to create and maintain data for TOU maps, you may define TOU map templates, which can be used to generate data for a TOU map. The templates may be used to define standard data for a TOU map as well as data for special periods, such as holidays. TOU map types reference TOU map templates.

Every TOU map references a TOU map template that defines the rules for generating TOU data from that TOU map. Specifically, TOU map templates define:

- The TOU group (defines the valid TOU periods for the template) used for the TOU map
- The default TOU period used for periods not explicitly defined. (This means you don't have to specify dates and times for all periods. For example, if your default TOU period is "Off Peak" you only need to define dates and days and times for On Peak or other TOU periods.)
- The specific date ranges, days of the week, and time periods designated for each TOU period. The system periodically generates TOU map data for TOU maps by interpreting the rules defined in the template.

Holidays

Many utilities categorize consumption on holidays differently than on the day of week on which the holiday falls. For example, holiday consumption might be categorized as Off-Peak regardless of the day it falls on. TOU map templates can define rules for different TOU periods for holidays by specifying the following:

- A Work Calendar that defines when holidays start and end
- Either:
 - A Holiday TOU period for consumption on holiday
 - A Holiday TOU Map Template that defines the TOU codes to use for different times in the year

TOU Map Template Interval Size

TOU map templates can also specify an interval size. This value specifies the duration of the individual TOU map data records, and also controls the values allowed in the Start and End Times. For example, if a TOU map template sets the interval size at 15 minutes, Start and End times must be in units of the interval size (10:00, 10:15, 10:30, etc.).

A TOU map template can be used to generate TOU map data for TOU maps whose interval size is divisible by the template's interval size. For example, a 60 minute template can be used to generate TOU data for TOU maps with an interval size of 60 minutes, 15 minutes, 5 minutes, etc. This means separate map templates are not needed for every interval size.

Refer to [Creating New Style TOU Map Templates](#) for further information.

Setting Up Time of Use (TOU) Mapping and Pricing Control Tables

This section provides an overview of setting up your control tables to support time of use mapping and pricing with rates.

TOU Codes

Refer to [Setting Up Time of Use Codes](#) for further information

TOU Groups

Refer to [Setting Up TOU Groups](#) for further information

Creating New Style TOU Map Templates

TOU map templates may be used to define standard data for a TOU map as well as data for special periods, such as holidays. TOU map types reference TOU map templates and used for TOU map data generation.

Prerequisites: You must define TOU groups and work calendars before you can create TOU map templates. Refer to the Oracle Utilities Application Framework online help for more information about creating work calendars.

To maintain existing TOU map templates, select **Admin > Consumption > TOU Map Template > Search** then use standard actions to edit, duplicate, or delete a TOU map template.

To define a new TOU map template, follow these steps:

1. Select **Admin > Consumption > TOU Map Template > Add**. If your system supports more than one TOU map template business object, you will be prompted to select a business object for this TOU map template.
2. Enter a name and a meaningful description for the TOU map template.
3. Select the TOU group to be used by the TOU map template.
4. Select the default TOU for the TOU map template (from the TOU Group). This is the TOU used when creating TOU map data for dates not accounted for in the TOU Schedules section.
5. Select the work calendar for the TOU map template. Work calendars define the days of the week on which work is performed, and specify holidays.
6. Select the holiday TOU for the TOU map template (from the TOU Group).
7. Select the TOU map template used for holidays (if applicable).
8. Specify the interval size for TOU map data created from the map template. Interval size is designated as hours:minutes:seconds (HH:MM:SS)
9. To specify TOU schedule's date ranges and which TOUs should be used for this TOU map template, click the + or - sign in the TOU Schedule Section and enter or select the following:
 - Start and End Dates for a specific date range
 - Start and End Days of the Week: To add or remove Start and End Days of the Week pairs, click the + or - sign and select the appropriate weekdays
 - Start and End Times: To add or remove Start and End Times within a Start and End Days of the Week pair, click the + or - sign and enter the appropriate times
 - TOU
10. Click **Save**.

Now you can use the TOU map template when creating TOU types.

Creating New Style TOU Map Types

An interval data curve is mapped into time-of-use periods using a TOU map. The TOU map used by the calculation rule for mapping interval data curve values to time of use periods is defined directly on the calculation rule. TOU map types define attributes shared by TOU maps of a given type.

Prerequisites: You must define TOU map templates and time zones before you can create TOU map types. Refer to the Oracle Utilities Application Framework online help for more information about creating time zones.

To maintain existing TOU map types, select **Admin > Consumption > TOU Map Type > Search** then use standard actions to edit, duplicate, or delete a TOU map type.

To define a new TOU map type, follow these steps:

1. Select **Admin > Consumption > TOU Map Type > Add**.

NOTE: If your system supports more than one TOU map type business object, you will be prompted to elect a business object for this TOU map type.

2. Enter a name and a meaningful description for the TOU map type.
3. Select the business object to use when creating TOU maps of this type

4. Select the time zone for the TOU map type.
5. Specify the interval size for TOU map data created from the map type. Interval size is designated as hours:minutes:seconds (HH:MM:SS)
6. Select the default TOU map template for the TOU map type
7. To add or remove override TOU map templates for this TOU type, click the + or - sign in the Override TOU Map Templates section and select the TOU map template.
8. Click **Save**.

Now you can use the TOU map type when creating TOU maps.

Defining Premise & Service Point Options

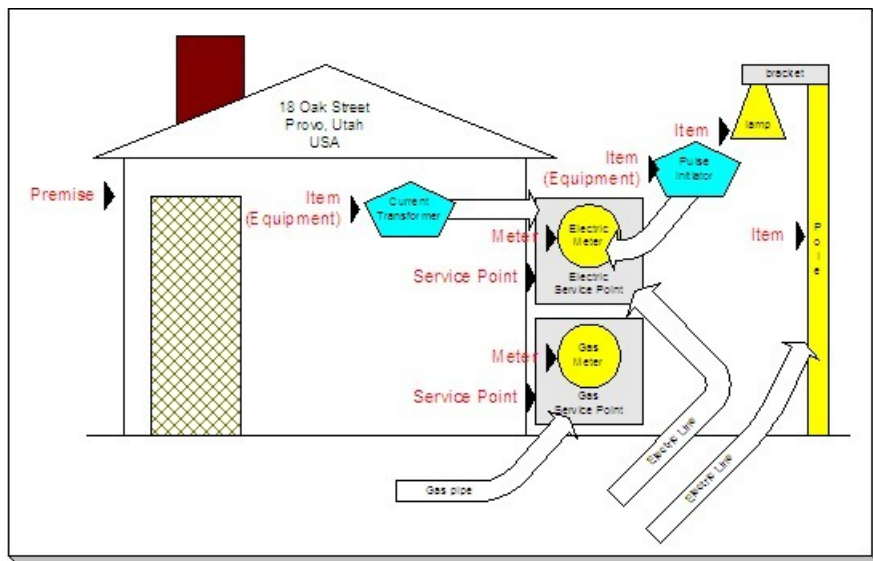
A premise is where a customer consumes the services supplied by your company. Located at a premise are the various devices that consume energy and measure consumption. Before you can define premises and devices, you must set up the control tables defined in this section.

WARNING:

The topics in this section do not describe every table that must exist in order to set up premises and service points. Many premise and service point control tables are described in [Defining Meter & Item Options](#), and [Defining Field Order Options](#). The tables described in this section are those that must be set up regardless of the type of service.

An Illustration Of A Premise

The following picture illustrates a premise with 2 service points, 2 meters, and 2 badged items:



The following concepts are illustrated above:

Premise A premise describes a location at which your company supplies some type of service. In addition to the obvious address information, a premise also contains geographic coordinates, meter read instructions, and taxation jurisdiction information.

FASTPATH:

For more information about the control tables that must be set up before you can define a premise see [Setting Up Premise Options](#).

Service Point A service point (SP) is a geographic location at which service(s) are delivered to a premise. The SP record maintains information about the type of service, the service cycle (if the service is metered), the field office responsible for maintaining the service, the distribution company that supplies the service, etc.

There are three major categories of service points:

- Those where the rate of consumption and the total amount of consumption is measured (e.g., electricity, gas, water) by a meter. You can think of this type of service point as a "socket" into which a meter can be plugged. Over time, many meters may be plugged into the socket. We refer to these types of service points as metered.
 - Those that hold badged items. A badged item is a physical device with a unique identity (e.g., a specific street light, a specific hydrant). You can think of this type of service point as a "socket" into which a badged item can be plugged. Over time, many items may be plugged into the socket. We refer to these types of service points as item-based.
 - Those used to hold one or more non-badged items. For example, if your organization doesn't badge street lamps, you can use a single service point to hold an infinite number of lamps. We refer to these types of service points as non-badged.
-

FASTPATH:

Refer to [Service Points \(SPs\)](#) for more information about non-badged items.

An unlimited number of SP's may exist at a premise. However in reality, the number of SP's is related to the number of services supplied by your company. For example, an electric and gas company will typically have 2 SP's per premise.

FASTPATH:

For more information about the control tables that must be set up before you can define service points refer to [Setting Up Generic Service Point Options](#), [Defining Meter & Item Options](#).

Field activities may be dispatched to all types of service points.

Meter A meter is a physical device used to measure the amount of gas, water, or electricity used by a customer. While most meters measure consumption in a single unit of measure (e.g., gallons, cubic feet, kilowatt-hours), some electric meters are extremely sophisticated and measure several different factors. For example, some electric meters measure how much was used, when it was used, the efficiency of consumption, the maximum amount used, and a few other unusual things.

FASTPATH:

For more information about the control tables that must be set up before you can define a meter, refer to [Setting Up Meter Options](#), and [Setting Up Consumption Estimation Parameters](#).

Item (Equipment) An item that is considered to be "equipment" is a physical device that regulates consumption; it does NOT measure consumption. You would only define equipment if it is of interest to your organization. For example, if your organization periodically tests the pulse initiators associated with your meters, you will need to set up items for each pulse initiator and link them to their respective meters. Equipment can be linked to either a service point (e.g., a current transformer, a backflow device), a meter (e.g., a pulse initiator), or an item (e.g., the components of an installation).

NOTE:

Equipment and billing. Be aware that the only way equipment can impact billing would be if you developed pre-processing calculation groups that analyzed the equipment associated with a service point (directly or indirectly via the meters and items) and manipulated billed consumption accordingly. Refer to [Understanding Calculation Groups and Rules](#) for more information.

Item (NOT Equipment) An item that is not consider to be "equipment" is a physical device that does NOT measure consumption, but impacts billing in some way (i.e., there are charges in your rates based on the number and type of items installed at a service point). Examples include street lights, light poles, and security cameras. Items are related to service points and a service point can have one or more items linked to it.

FASTPATH:

For more information about the control tables that must be set up before you can define an item, refer to [Setting Up Item](#).

For more information about premises and service points, refer to [Understanding The "V"](#).

Setting Up Premise Options

This section describes tables that must be set up before you can define premises.

Defining Premise Types

Open **Admin > Geographic > Premise Type** to define the premise types used to categorize your premises.

Description of Page

Enter a unique **Premise Type** and a **Description** for every premise type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_PREM_TYPE](#).

Implementing Address Validation

In order to set up address validation, you must ensure that the following Option Types are defined with their corresponding values on the General System Configuration – Feature Configuration page:

Option Type	Description
Address Validation Script	Indicates the name of the BPA script to invoke when a user clicks on the validate address button. This option is required if the Allow Address Validation option is set to Y. The base product includes sample script C1-ValAddr that uses Oracle Spatial procedures and functions to geocode an address and return its matching quality code.
Allow Address Validation	Indicates whether an integration with address validation software is implemented. If set to Y, the system will render a validate button along with the address fields displayed throughout the application to trigger address validation. Valid values are 'Y' and 'N'. If set to Y, the Address Validation Script to invoke must be specified.

As described in the above table, the base product includes sample script C1-ValAddr uses Oracle Spatial procedures and functions to geocode an address and return a matching quality code. To use this script, ensure that the Oracle Spatial Geocoding (F1-ORAGEOCD) algorithm is plugged in on installation options for the geocoding service system event. A data source containing the geocoding data must also exist on the database.

If your implementation uses something other than Oracle Spatial for address validation, you'll need to create a new address validation BPA script to specify on the General System Configuration feature configuration. For additional information about creating this script, see the steps descriptions for the C1-ValAddr script in the application. The new script should contain the following move step to update the page:

- If the calling page is an old style page, the move from the BPA to the page can be achieved by specifying a move step from a source field to a User Interface Field.

- If the calling page is a display map, the move from the BPA to the page can be achieved by specifying a step to read the Business Object, modify the values as returned by the validation software, and then update the BO.

Setting Up Generic Service Point Options

This section describes tables that must be set up before you can define service points.

Facility Levels

Every type of service tends to use a different mapping philosophy to designate the facility hierarchy that supplies service to the service point. For example, electric service typically uses a substation / feeder / node facility hierarchy to define how electricity is supplied to a service point (the substation is the highest level in the hierarchy, the feeder comes next, and finally the node). Whereas gas service uses a city gate / main / feeder hierarchy.

If your organization maintains this type of information on service points, you will set up your facilities and their interrelationships using 3 windows. On the first you set up the number and type of facility levels used for every service and you define the valid values for each facility level (you define these when you define your Service Types). On the second and third you define the values that may coexist in each level. After these set up tasks are complete, you're ready to enter facility levels on your service points.

NOTE:

A service point's facility levels are used to help pinpoint problems and dispatch service crews during outages.

Setting Up Facility Levels 1 & 2 Combinations

To define which values of facility level 2 may be used with a given value of facility level 1, open **Admin > Geographic > Facility Level 1 to 2**.

Description of Page

Choose a **Service Type** and **Facility Level**, then use the **Facility Level 2** collection to define the level 2 facility levels that may coexist with the selected level 1 facility level.

Setting Up Facility Levels 2 & 3 Combinations

To define which values of facility level 3 may be used with a given value of facility level 2, open **Admin > Geographic > Facility Level 2 to 3**.

Description of Page

Choose a **Service Type** and **Facility Level 2**, then use the **Facility Level 3** collection to define the level 3 facility levels that may coexist with the selected level 2 facility level.

Setting Up Service Point Types

Every service point must reference a service point (SP) type. The SP type controls almost all aspects of the service point's behavior (e.g., the type of field activity that may be dispatched to it, the type of service agreement that may be linked to it, the type of meter that may be installed at it).

The topics in this section describe the windows used to set up your SP types.

WARNING:

Setting up SP types requires careful analysis of your company's SA types, field activities, and its consumption estimation philosophy. Refer to [Designing SP Types](#) for more information about this design process.

SP Type - Main

You begin to define a service point type by selecting **Admin > Geographic > SP Type > Add.**

WARNING:

You may find that your desire to use some of the more sophisticated control functions in the system will necessitate many SP types. For example, notice that one of the other windows on this group allows you to define the meter types that can be installed in service points of a given type. If you have many different types of meters and many restrictions as to the types of service points in which they can be installed, you will end up with many SP types.

Description of Page

Enter an easily recognizable **SP Type** for the service point type. This value will appear on many windows throughout your system as a "shorthand" for how the SP is used, so think carefully about the format. We recommend using a consistent format for all of your SP types. You should include the service type, an indication if it's metered or item-based, and the trend classification. For example, you might enter "E-M-RES", for an electric, metered, residential SP type.

Select the **Service Type**. Refer to [Setting Up Service Types](#) for more information.

Enter a **Description**.

Select the **Sub Type** to indicate the type of device that may be installed at service points of this type: Meter, Item, Unbadged.

Turn on the **Allow Service Route** switch if service points of this type should have a route. This switch would typically be turned on for metered service points.

Multiple Route Usage controls whether a service point can reference multiple service routes. Select Allowed if multiple service routes are allowed on service points of this type. Select Not Allowed if multiple service routes are not allowed on service points of this type. This switch would typically be turned on for garbage service points.

If this SP type is for metered service, select the **Trend Class** to define how to categorize this SP type's consumption for estimation and high / low validation purposes. This field will be gray for item-based services because items don't have meter reads that can be estimated or that are subject to high / low validation. Refer to [Setting Up Trend Classes](#) for more information.

If this SP type is for fieldwork that is managed by an external system, specify the external system in **Fieldwork Orchestration**.

If field activities are created for this SP type's service point, select the **Field Activity Type Profile** that controls which type of field activities may be linked to the service points. This Profile will also control which field activities are automatically generated by the system under various circumstances. Refer to [Setting Up Field Activity Type Profiles](#) for more information.

If a geographic type is used to identify individual service points of this SP type, select the **Identifying Geographic Type** used. You typically have an identifying geographic type if you refer to a service point using an identifier that is assigned by a third party. Refer to [Defining Geographic Types](#) for information on setting up geographic types.

Use the **SP Type Characteristic** collection and their respective **Characteristic Values** to describe characteristics common to all service points of this type.

NOTE:

You can only choose characteristic types defined as permissible on an SP Type record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SP_TYPE](#).

SP Type - Meter Type

Open **Admin > Geographic > SP Type > Search** and navigate to the **Meter Types** page to define the types of meters that may be installed at service points belonging to this SP type.

Description of Page

The **Meter Types** collection contains the types of meters that may be installed at service points belonging to this SP type. This collection is not relevant if the SP sub type is Item or Unbadged.

NOTE:

You can connect meters with different service types to your SP Type. For example, a metering device such as a recorder can be used on both gas and electric service points.

SP Type - SA Type

Open **Admin > Geographic > SP Type > Search** and navigate to the **SA Types** page to define the SA types that may be linked to (and therefore pay for) service points belonging to this SP type.

Description of Page

The following fields display:

CIS Division / SA Type Indicate the type of service agreement that may be linked to service points of this type.

Initial Turn on this switch if the Start Service process should default this SA type when service is initially started at a service point of this type. Multiple SA types may be marked as Initial if you want multiple service agreements created when service is initially started. For example, if you have both wastewater and water service agreements linked to the water service point.

SP Type - Item Type

Open **Admin > Geographic > SP Type > Search** and navigate to the **Item Types** page to define the types of items that may be installed at service points belonging to this SP type.

Description of Page

The **Item Types** collection contains the types of items that may be installed at service points belonging to this SP type. This collection is not relevant if the SP sub type is Meter or Unbadged.

NOTE:

You can connect items with different service types to your SP Type. For example, an item like a cell phone can be used on both gas and electric service points.

SP Type - SP Characteristic

To define characteristics that may be defined for service points of a given type, open **Admin > SP Type > Search** and navigate to the **SP Characteristic** page.

Description of Page

Use the **SP Characteristics** collection to define characteristics that can be defined for service points of a given type. Turn on the **Required** switch if the **Characteristic Type** must be defined on service points of a given type. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

SP Type - Equipment Type

Open **Admin > Geographic > SP Type > Search** and navigate to the **Equipment Types** page to define the types of equipment that can be linked to service points of a given type.

Description of Page

Use the collection to define the item types of **Equipment** that can be linked to service points of this type.

NOTE:

Item types are being specified. There is no equipment type control table. Rather, items are used to define equipment and therefore you are actually defining item types rather than equipment types. Refer to [Equipment versus Badged Items](#) for more information about equipment.

SP Type - FA Type Review

Open **Admin > Geographic > SP Type > Search** and navigate to the **FA Type Review** page to review the field activities that are allowed for the SP type.

NOTE:

Four dimensions. For every **field activity type** that is eligible for dispatch, you define the **dispatch group** that performs the activity at every **SP type** located in every **operations area**. This information is maintained on the field service control page. This is a rather complex relationship because it involves the four dimensions highlighted in bold. Due to this complexity, we have provided review trees on the SP Type, Dispatch Group, and Field Activity Type windows to help you understand what you've set up.

Description of Page

This window is dedicated to a tree that shows the field activities allowed for this SP type. For each field activity type that is eligible for dispatch, you can view the dispatch group that will perform the activity at every operations area.

Setting Up Premise & Service Point Postal Defaults

You set up postal defaults if your company is able to default field values onto new premises and service points based on the premise's postal code. The topics in this section describe how to maintain postal defaults.

FASTPATH:

For more information about where these default values are used, refer to [Maintaining Premises](#) and [Maintaining Service Points](#).

Postal Defaults - Main

To define premise postal defaults, open **Admin > Postal Code Default > Geographic > Add**.

Description of Page

Enter the **Country Code** and range of postal codes to which the default values apply using the **From Postal Code** and **To Postal Code**.

NOTE:

You may not have postal defaults whose from / to postal codes overlap.

Select the **Trend Area** to be defaulted onto new premises located in this postal code range. The trend area is used to categorize premises into geographic areas when consumption estimation is controlled, in part, by where the customer lives. Refer to [Setting Up Trend Areas](#) for more information.

Enter the **County** to be defaulted onto new premises located in this postal code range.

Enter the **City** to be defaulted onto new premises located in this postal code range.

Enter the **CISDivision** to be defaulted onto new premises located in this postal code range.

Enter the **State** to be defaulted onto new premises located in this postal code range.

Enter the **Time Zone** to be defaulted onto new premises located in this postal code range.

Use the **Characteristic Types and Values** collection to define the **Characteristic Types** and their respective **Characteristic Values** to be defaulted on premises located in this postal code range. In addition to providing interesting information, these characteristics may also determine the prices and tax rates on the bills generated for the services consumed at a premise.

FASTPATH:

For more information about characteristics, see [Setting Up Characteristic Types & Their Values](#) and [An Illustration Of A Bill Factor And Its Characteristics](#).

Use the **Geographic Types and Values** collection to define the **Geographic Types** and their respective **Values** to be defaulted on premises located in this postal code range. In addition to providing interesting information, these values may be used to sort field activities in geographic value order.

Where Used

This information is defaulted when a new Premise is added. Characteristics and geographic values are also defaulted when the postal code for a Premise is changed. Refer to [Maintaining Premises](#) for more information.

Postal Defaults - Service Default

To define values to be defaulted for a service point located in a postal range, open **Admin > Postal Code Default > Search** and navigate to the **Service Defaults** page.

FASTPATH:

For more information about where these default values are used, refer to [Maintaining Service Points](#).

Description of Page

Use the **Service Defaults** collection to define values to be defaulted on service points located in a given postal code range belonging to a given **Service Type** (note, a service point's Service Type comes from its SP Type).

The following values may be defined per **Service Type**:

- When a meter or item is removed from a service point located in this postal code range, the system assigns it by default to **Stock Location**. Note that this is only used if the meter's meter type or item's item type indicates that stock locations are tracked.

FASTPATH:

For more information about location history, refer to [Stock Location History Is Created Behind The Scenes](#).

- Use the **Field Services** collection to define the **Field Service Classifications** and their respective **Operation Area** to be defaulted on new service points located in this postal range. Refer to [Designing Field Service Classifications](#) for more information.
- Use the **Characteristic Types and Values** collection to define the **Characteristic Types** and their respective **Characteristic Values** to be defaulted on new service points located in this postal code range. In addition to providing interesting information, these characteristics may also determine the prices and tax rates on the bills generated for the services consumed at a service point.
- Use the **Geographic Types and Values** collection to define the **Geographic Types** and their respective **Values** to be defaulted on new service points located in this postal code range. In addition to providing interesting information, these values may be used to sort field activities in geographic value order.

Where Used

Information is defaulted when a new Service Point is added to a Premise. Refer to [Service Point - Main Information](#) for more information.

Stock Location is defaulted when a Meter or an Item is removed from a Service Point. [Refer to SP/Meter Installation](#) and [SP/Item Installation](#).

Designing SP Types

Every service point must reference an SP type. When you set up an SP type, you define how the system manages many aspects of its service points' behavior.

NOTE:

Perfect foresight. In a perfect world, the other control tables would have been set up with perfect foresight of setting up your SP types. In reality, setting up your SP types may invalidate some of your earlier decisions. Don't feel bad if this happens, some amount of iteration is natural.

Designing your SP types is an iterative process. To minimize the number of iterations, we recommend using the steps outlined in this section to complete the following table. When the table is complete, you're ready to set up your SP types.

SP Type	Service Type
---------	--------------

Service Segmentation

At a minimum, you will have one SP type for every different type of service that exists at your premises. If we assume that your organization sells gas, water, waste water and electricity services, you will need four SP types.

SP Type	Service Type
GAS	Gas service
WATER	Water service
WASTE WATER	Waste water service
ELECTRIC	Electric service

Device Segmentation

For each service, determine if there exist meters, badged items, or non-badged items at the service points. For example, if we assume:

- Electric service has service points with meters, badged lamps, and parking lots.
- Gas and water services just use meters.
- Waste water service doesn't actually have a meter installed at it (it uses the water consumption measured by the water service point's meter)

Then your SP types will be:

SP Type	Service Type	SP Sub Type
GAS - METERED	Gas service	Meter
WATER - METERED	Water service	Meter
WASTE WATER	Waste water service	Unbadged
ELECTRIC - METERED	Electric service	Meter
ELECTRIC - BADGED LAMP	Electric service	Item
ELECTRIC - PARKING LOT	Electric service	Unbadged

Meter Read Estimation Trend Class Segmentation

When you set up a metered SP type, you must define the consumption estimation trend class in which its consumption will be categorized. This categorization matters when consumption differs based on the type of property AND you want the system to estimate consumption in different classes differently.

These categories could be the classic divisions of residential versus commercial versus industrial consumers. Alternatively, they could be finer-grained divisions: single family residence, versus duplex, versus triplex, versus medical office, versus grocery store.

FASTPATH:

Refer to [The Theory Behind Consumption Estimation](#) for more information about how trend class is used to estimate consumption.

If we assume you differentiate between residential and commercial service for all of your metered services, then your SP types will be:

SP Type	Service Type	SP Sub Type	Trend Class
GAS - METERED - RESID	Gas service	Meter	Residential
GAS - METERED - COMM	Gas service	Meter	Commercial
WATER - METERED - RESID	Water service	Meter	Residential
WATER - METERED - COMM	Water service	Meter	Commercial
WASTE WATER - RESID	Waste water service	Unbadged	N/A
WASTE WATER - COMM	Waste water service	Unbadged	N/A
ELECTRIC - METERED - RESID	Electric service	Meter	Residential
ELECTRIC - METERED - COMM	Electric service	Meter	Commercial
ELECTRIC - BADGED LAMP	Electric service	Item	N/A
ELECTRIC - PARKING LOT	Electric service	Unbadged	N/A

Notice that the non-metered service points don't use a trend class. This is because they don't have meters (and only meters have estimated consumption).

Field Activity Type Profile Segmentation

When you set up any type of service point that can have field activities, you must define the field activity type profile. This profile defines:

- The type of field activities that may be dispatched to the service points.
- The type of field activity to be defaulted by the start / stop process given the condition of the service point.

Field activity type profiles should not impact your SP type design as these profiles should be designed after the SP types are designed.

FASTPATH:

Refer to [Designing Your Field Activity Profiles & Types](#) for more information about how field activity profiles are used.

SA Type Segmentation

Every SP Type whose service points can be linked to a service agreement has one or more SA types. These define which type of service agreements can pay for the service point's service. If different service points have different valid SA types, you will need to split the SP types accordingly.

The SA Type segmentation of SP Types is the most complicated design decision you'll have to make. Unfortunately, the decision making process is subjective and iterative. The iterations are caused by the fact that the number of SA types is dependent on the number of SP types (and vice versa). We recommend the following to help work your way through this conundrum:

- Design your SA types using the information in [Defining Service Agreement Types](#).

- Return to your SP types and determine if, given the proposed SA types, you can define a list of valid SA types for each SP type. If you find the population of SA types (and their valid rates) could result in invalid rates paying for service at a service point, divide your SA types further.

Meter Type Segmentation

Every metered SP Type has one or more meter types. These define which type of meter can be installed at a service point of a given type. If different service points have different valid meter types, you will need to split the SP types accordingly.

For example, if you have 3-phase electric service points and you want to make sure that only 3-phase meters are installed in these service points, you will need to split the electric SP types accordingly.

FASTPATH:

Refer to [Setting Up Meter Types](#) for more information.

Item Type Segmentation

All SP Types may have item types. These define which type of item can be installed at a service point of a given type. If different service points have different valid item types, you will need to split the SP types accordingly.

FASTPATH:

Refer to [Setting Up Item Types](#) for more information.

Defining Bill & Meter Read Cycles

This chapter is dedicated to issues related to defining cycles, routes and schedules in the system.

Defining Bill & Service Cycles

Every account references a bill cycle. An account's bill cycle controls when it is billed.

Every metered service point references a service cycle. A service point's service cycle controls when a service point's meter is read.

The design of your meter read and bill cycles is inextricably linked because you probably want to bill your customers shortly after their meters are read.

In this section, we describe how to design and set up these cycles. In addition, we discuss how to set up bill period schedules. These are used to define the bill segment end date for special types of non-metered service agreements.

NOTE:

Recommendation. We recommend reading [Bill Frequency - Bill Cycle vs Bill Segment Duration](#) before setting up this information.

The Big Picture Of Bill Cycles, Service Cycles and Bill Periods

The topics in this section provide background information about a variety of bill cycle, service cycle, and bill period issues.

Designing Cycles for Metered Services

The topics in this section provide background information about a variety of service cycle issues.

A Description Of The Cyclical Meter Read Process

Meter readers using handheld devices record most meter reads in the field. These meter reads are uploaded into the system for use by billing.

A service point's meter is read due to the following data relationships:

- Every metered service point references a service route (henceforth called "route").
- A route references a service cycle.
- A service cycle has service schedules that define when the service points in the cycle are read.
- Every service cycle schedule contains two dates:
 - **Scheduled selection date.** This is the date the system selects the service points in the cycle for download to your handheld software.
 - **Scheduled work date.** This is the date the meter is expected to be read.

NOTE:

Overriding a route within a specific schedule. Rather than downloading all routes within a cycle, you can set up the system so that only specific routes are downloaded on any given date. This is a very powerful feature. You can use it, for example, to estimate specific cycles every other month or indicate the customer reads the meter every third month. Refer to [Designing Service Cycles, Routes, And Schedules](#) for more information.

- On the scheduled selection date, the system creates a download file containing information to be sent to your handheld software. This download file contains information about every register on every meter in the routes being downloaded.
 - Your handheld software distributes this information to the handheld devices and then your meter readers do their job. When they finish, the resultant meter reads are uploaded into the system for subsequent use by billing.
-

WARNING:

It is very important to create a service cycle schedule for every expected read date regardless of whether the cycle's routes are downloaded. Why do you have to do this? Because billing uses the scheduled work date on the service cycle to know when to look for a reading. If it can't find a reading on or near this date, billing estimates consumption (given estimation is allowed on the service agreement). Without a service cycle schedule, billing wouldn't know when to look for readings. So, for example, if you estimate a given cycle's consumption EVERY OTHER MONTH, you must create a service cycle schedule for EVERY month. On each month's schedule, you must define if the routes should be downloaded.

FASTPATH:

For more information about the how to control when a cycle is read, refer to [Setting Up Service Cycle Schedules](#).

Designing Service Cycles, Routes, And Schedules

The topics in this section provide information describing how to design your service cycles, routes, and schedules.

- [Designing Service Cycles For Meter Reading](#)
- [Designing Service Routes For Meter Reading](#)
- [Designing Metered Service Cycle Schedules](#)

Designing Service Cycles for Meter Reading

The criterion that affects the number of service cycles has nothing to do with when meter readers physically read your meters. Rather, the frequency that you bill the meter's consumption (real or estimated) is what controls the number of service cycles.

So, for example, if you bill every month, but read every OTHER month, you'll have 20 service cycles - one for each bill day during the month. If you bill bimonthly, you'll have 40 service cycles. If you bill quarterly, you'll have 60 service cycles. Etc.

NOTE:

Different billing frequencies are possible for different service points. If you have different billing frequencies for your different types of metered service, you'll need a different set of cycles for each billing frequency. For example, if you bill water quarterly and electricity monthly, you'll have 20 bill cycles (one for each bill day during a month), but you'd need 60 quarterly service cycles for your water service points, and 20 monthly service cycles for your electric service points. This would result in a customer getting billed every month. However, four times a year, their bill would contain a charge for both electricity and water.

[Top of the Page](#)

Designing Service Routes For Meter Reading

The following points describe the relationship between a meter read, a route and a service cycle:

- A service cycle contains one or more routes.
- A route has one or more service points.
- A service point holds a meter.
- And a meter is what is read.
- Therefore, the number of meters a person can read in a day limits the number of service points in a route.

WARNING:

If your company supplies electric service and uses MV90's, you will need to take advantage of "notional" routes. A "notional" route's service points are never actually read by a human. Rather, the service points' consumption is fed to the system by a sophisticated device (e.g., an MV90). These service points still need to be linked to a route because billing is dependent on the route's cycle to determine the expected meter read date.

[Top of the Page](#)

Designing Metered Service Cycle Schedules

The process of designing your service schedules is either easy or complicated. It will be easy if all routes within a cycle are downloaded when the service cycle is scheduled for download. It will be complicated if you download a subset of routes within a cycle on any given download date. We'll provide a few examples to help explain why.

If you download all cycles in a route whenever the cycle is downloaded, your service cycle schedule will look as follows.

NOTE:

Bill cycles. We've included each service cycle's related bill cycle to help you understand when the service cycle's consumption will be billed. Bill cycles are discussed in [Bill Cycles](#).

Service Cycle	Download Date	Sched MR Date	Which Routes To Download	Bill Cycle	Bill Window	Estimation Date
1	28-May-99	31-May-99	All	1	31-May-99 to 2-Jun-99	2-Jun-99
2	31-May-99	1-Jun-99	All	2	1-Jun-99 to 3-Jun-99	3-Jun-99
3	1-Jun-99	2-Jun-99	All	3	2-Jun-99 to 4-Jun-99	4-Jun-99
4	2-Jun-99	3-Jun-99	All	4	3-Jun-99 to 7-Jun-99	7-Jun-99
Etc. to 20						

Now let's complicate life. If we assume you physically read your routes every other month (but bill monthly using estimated consumption), then you'll need the following service schedule.

Service Cycle	Download Date	Sched MR Date	Which Routes	Bill Cycle	Bill Window	Estimation Date
1	30-May-99	31-May-99	1, 2, 3 - Download 4, 5, 6 - Estimate	1	31-May-99 to 2-Jun-99	2-Jun-99
2	31-May-99	1-Jun-99	1, 2, 3 - Download 4, 5, 6 - Estimate	2	1-Jun-99 to 3-Jun-99	3-Jun-99
3	1-Jun-99	2-Jun-99	1, 2, 3 - Download 4, 5, 6 - Estimate	3	2-Jun-99 to 4-Jun-99	4-Jun-99
4	2-Jun-99	3-Jun-99	1, 2, 3 - Download 4, 5, 6 - Estimate	4	3-Jun-99 to 7-Jun-99	7-Jun-99
Etc. to 20						
1	29-Jun-99	30-Jun-99	1, 2, 3 - Estimate 4, 5, 6 - Download	1	30-Jun-99 to 2-Jul-99	2-July-99
2	30-Jun-99	1-July-99	1, 2, 3 - Estimate	2	1-July-99 to 3-Jul-99	3-July-99

			4, 5, 6 - Download			
3	1-July-99	2-July-99	1, 2, 3 - Estimate	3	2-July-99 to 4- July-99	4-July-99
			4, 5, 6 - Download			
4	2-July-99	3-July-99	1, 2, 3 - Estimate	4	3-July-99 to 7- July-99	7-July-99
			4, 5, 6 - Download			
Etc. to 20						

Notice the following:

- You still have 20 service cycles even though you only read the meters every other month. Why? Because billing uses the scheduled read date on the service cycle to know when to look for a reading. If it can't find a reading on this date, billing estimates consumption (given estimation is allowed on the service agreement). Without a service cycle schedule, billing wouldn't know when to look for readings.
- Every other month you download half the routes in each cycle and estimate consumption for the other half.
- If you don't download all routes when a service cycle is scheduled, you have to indicate how to handle every route in the cycle.

The above design is infinitely flexible. You can use it to handle any number of requirements:

- Estimate consumption every seventh month.
- Bill every month, but only read once a quarter.
- Etc.

Designing Cycles for Waste Collection Services

The topics in this section describe how to design service cycles to support your waste collection requirements.

A Description Of The Cyclical Waste Collection Process

Waste collection equipment (e.g., garbage trucks) travel from service point to service point using a predefined schedule. This schedule defines which groups of service points are collected by a given truck on a given day (we use the term "route" to define a group of service points that are collected by a given truck on a given day).

Waste service points tend to belong to multiple routes because they can be collected in a different order depending on a wide variety of factors:

- A service point may have one truck responsible for collecting recycling and a different truck responsible for collecting non-recyclable waste.
- A service point that is collected on Mondays, Wednesdays and Fridays may be on a different route on each day of the week.
- Etc.

NOTE:

Contrary to metered service points, the service schedules linked to your waste collection service cycles / route groups do not affect billing. Why? Because billing doesn't need to look for a read for these types of service points and therefore it doesn't need a schedule to know when to look for a read.

Bill Cycles

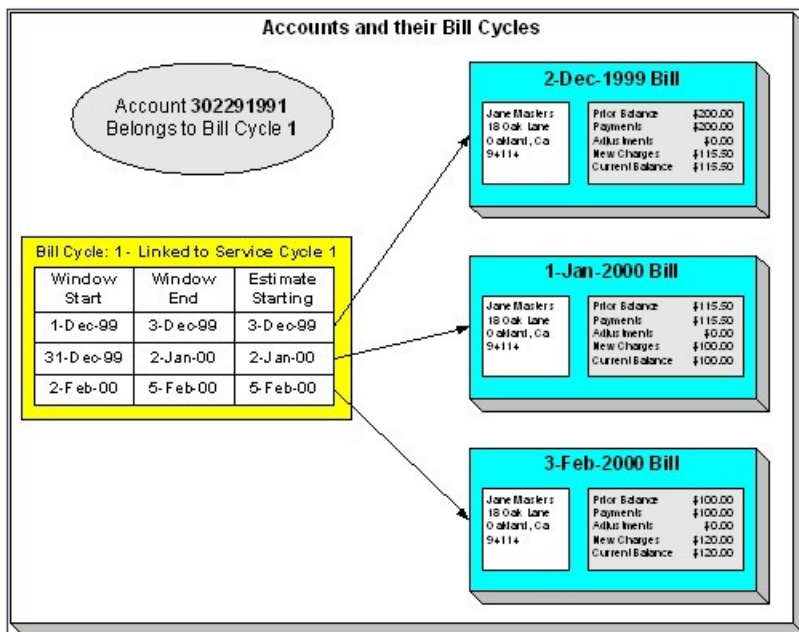
The topics in this section provide background information about a variety of bill cycle features.

The Cyclical Billing Process & Window Billing

The cyclical bill creation process creates most bills. This process works as follows:

- Every account references a bill cycle. The bill cycle's schedule controls WHEN the system attempts to create bills for the account.
- Every bill cycle has a bill cycle schedule that defines the dates when a cycle's accounts are to be billed. Rather than attempt to create bills on one evening, most bill cycles use a concept of "window billing" where the system attempts to produce bills for accounts over a few nights. The first night (i.e., the day the window opens) should be the earliest day on which meter reads for the account can enter the system. The last night (i.e., the day the window closes) should be the last possible day in which a meter read can enter the system. This concept of window billing allows you to start billing accounts on the earliest possible day and then bill the stragglers over successive evenings. This results in much better cash flow.
- When the bill cycle creation process runs, it looks for bill cycles with open bill windows. It then attempts to create bills for the accounts in each such cycle. If a bill is created, it will send it out that evening. If a bill cannot be created, the system will create a bill in the "error" state with a message that can be analyzed by your billing staff. The next day, your staff can either correct the problem or not. When the bill cycle creation process next runs, it deletes all "error" bills and attempts to recreate them. It continues this throughout the bill window. If bills are in error at the end of the window, they will remain in this state until a user fixes them. If the bills are still in error when the cycle's next window opens, a billing error will be generated.

The following diagram should help clarify the above.



FASTPATH:

For more information about the how to control when bills are produced for a cycle, refer to [Bill Cycle - Bill Cycle Schedule](#).

Designing Your Bill Cycle

The number of bill cycles is determined by the frequency that you bill your customers. So, for example, if you bill every month, you'll have 20 bill cycles - one for each bill day during the month. If you bill bimonthly, you'll have 40 bill cycles. If you bill quarterly, you'll have 60 bill cycles. Etc.

Keep in mind the following:

- You may need additional bill cycles if you allow customers to be billed off-cycle. For example, you could create a bill cycle called "Seniors" and link this to every senior citizen. You would set up this bill cycle's schedule to create bills shortly after social security checks are issued.
- You may need other bill cycles for customers with non-metered services (e.g., land leases, one time invoices).

IMPORTANT:

An account's bill cycle should attempt to create bill segments at least as often as the shortest service agreement duration. For example, if an account has both monthly and quarterly service agreements, the account should be placed on a monthly bill cycle. Refer to [Bill Frequency - Bill Cycle vs Bill Segment Duration](#) for more information.

The Relationship Between Metered Service Cycles and Bill Cycles

As you can deduce, a service point's service cycle is related to an account's bill cycle.

The following table is an example of how you would set up the various dates on the various schedules that control meter reading and billing.

Service Cycle	Download Date	Sched MR Date	Bill Cycle	Bill Window	Estimation Date
1	28-May-99	31-May-99	1	31-May-99 to 2-Jun-99	2-Jun-99
2	31-May-99	1-Jun-99	2	1-Jun-99 to 3-Jun-99	3-Jun-99
3	1-Jun-99	2-Jun-99	3	2-Jun-99 to 4-Jun-99	4-Jun-99
4	2-Jun-99	3-Jun-99	4	3-Jun-99 to 7-Jun-99	7-Jun-99
Etc.					

Notice the following about this example:

- The bill cycle code is the same as the meter read (MR) cycle. This is not necessary, it's just good practice.
- The bill window starts on the first date on which a meter read could be uploaded.
- The bill window ends on the day after the last possible day a read could be uploaded for the bill cycle.
- Billing is only allowed to estimate consumption on the last day of the bill window.

FASTPATH:

For more information about the bill cycle schedule, refer to [Bill Cycle - Bill Cycle Schedule](#). For more information about the service cycle schedule, refer to [Setting Up Service Cycle Schedules](#).

How Does An Account Get Its Bill Cycle?

Most accounts are created behind-the-scenes when a user uses the "add account" option on [Person - Main Information](#). An account created like this doesn't have a bill cycle. Rather, it sits in limbo awaiting the activation of its first service agreement. When a service agreement is activated, the system populates the account's bill cycle using the following algorithm:

- If an account's bill cycle is protected, the activation of a service agreement will not affect an account's bill cycle. Refer to [Protecting An Account's Bill Cycle](#) for more information.
- If the service agreement being activated is for metered service, the account is given a bill cycle that will generate bills shortly after the service point is read. The route the system follows to get this bill cycle is a bit indirect:
 - A metered service agreement references one or more metered service points.
 - Every metered service point references a service cycle (the service cycle controls when the meter at the service point is read).
 - Every service cycle references a default bill cycle. It is this bill cycle that is populated on the account paying for service at the service point.
- If the service agreement being activated is not metered, the system cannot populate the account's bill cycle because there is no service cycle from which to default the bill cycle. This means the account's bill cycle will be blank until a user specifies a bill cycle for the account (using the Account page).

NOTE:

A To Do entry highlights accounts without a bill cycle. A To Do entry highlights those accounts that require a user to specify a bill cycle. This entry is generated for accounts without a bill cycle that have active service agreements.

When the last service agreement linked to an account is stopped, the account's bill cycle will be changed so that the account will be final billed when billing next executes. Refer to [What Happens At Finalization Time?](#) for more information.

When a service point's service cycle is updated, and the account's bill cycle is not protected, the system automatically updates the account's bill cycle to be in sync with the service cycle. Note that this will only take place if the Allow Bill Cycle Synchronization **Option Type** on the General System Options [Feature Configuration](#) is set to Y.

Protecting An Account's Bill Cycle

Over time, as a customer moves from premise to premise, their bill cycle is changed behind-the-scenes to be in sync with the latest service point's service cycle (as described in the previous section). If you do not want the system to change an account's bill cycle when a metered service agreement is activated, you need to turn on the account's **protect bill cycle** flag. You would do this if a customer liked to be billed at the start of the month regardless of when their meter was read.

When the last service agreement for an account is stopped, the protect bill cycle flag is reset. This is to ensure that if the customer returns to start new service again, the bill cycle can be set based on [How Does An Account Get Its Bill Cycle](#).

What Happens If An Account Has Service Agreements Spanning Metered Service Cycles?

A single account can have service agreements that are in several service cycles. The bill cycle on such an account will default based on the last activated metered service agreement.

It's important to be aware that an account will only have bills created when its bill cycle schedule so indicates. This means that the consumption for some service points could remain unbilled for a few weeks.

Designing Bill Periods

Bill periods are used by non-metered service agreements whose bill end dates need to fall on strict dates. You need only set up this information if you have this type of service agreement.

FASTPATH:

Refer to [Ways To Control The End Date Of A Bill Segment](#) for more information.

Every bill period has a calendar that defines when bill segments are created for service agreements that use the bill period. Examples of bill periods include:

- Quarterly Bill - Last Day Of Quarter
- Quarterly Future Bill - Last Day Of Quarter
- Monthly - 15th Day Of Month.
- Monthly Future Bill - Last Day Of Month.

The Quarterly Bill - Last Day Of Quarter bill period would have a schedule that looked as follows:

Earliest Date On Which To Create A Bill Segment	Bill End Date
1-Oct-1998	30-Sep-1998
1-Jan-1999	31-Dec-1998
1-Apr-1999	31-Mar-1999
1-Jul-1999	30-Jun-1999
...	

The Quarterly Future Bill - Last Day Of Quarter bill period would have a schedule that looked as follows:

Earliest Date On Which To Create A Bill Segment	Bill End Date
15-Dec-1998	31-Mar-1999
15-Mar-1999	30-Jun-1999
...	

The remainder of this section provides examples using the above calendars.

The following example assumes the following:

- The service agreement starts on 18-Dec-1998.
- The service agreement's SA type references the Quarterly Future Bill - Last Day Of Quarter bill period.

The following table shows when bill segments will be produced (assuming the account's bill cycle attempt to create segments as soon as possible) for several bill periods:

Earliest Date Segment Will Be Produced	Bill Period
18-Dec-1998	18-Dec-1998 thru 31-Mar-1999

...

The following example assumes the following:

- The service agreement starts on 18-Dec-1998.
- The service agreement's SA type references the Quarterly Bill - Last Day Of Quarter bill period.

The following table shows when bill segments will be produced (assuming the account's bill cycle attempts to create segments as soon as possible) for several bill periods:

Earliest Date Segment Will Be Produced	Bill Period
1-Jan-1999	18-Dec-1998 thru 31-Dec-1998
1-Apr-1999	1-Jan-1999 thru 31-Mar-1999

...

FASTPATH:

Refer to [Setting Up Bill Periods](#) for information about how to define this information.

Setting Up Bill Cycles

An account references a bill cycle. The bill cycle defines when the account is billed and when the account's debt is checked to determine if it's overdue. To define a bill cycle and its bill cycle schedule, open **Admin > Billing > Bill Cycle > Add**.

Description of Page

Enter a unique **Bill Cycle** and **Description** for every bill cycle.

Use the **Bill Cycle Schedule** collection to define when bills are produced for the accounts in a given bill cycle. The following fields are required for each instance:

Window Start Date Specify the date on which the system should start trying to create bills for accounts in the cycle.

Window End Date Specify the last day on which the system will create bills for accounts in the cycle. This should be the last possible date on which a meter read could be loaded into the system for the account.

Accounting Date Specify the financial date associated with the bills' financial transaction. The accounting date defines the financial period(s) to which the bills will be booked in your general ledger.

Estimate Date The date on which the system is allowed to start estimating consumption if a real read cannot be found. Leave this date blank to inhibit consumption estimation. When specified, this date is typically on or shortly before the window end date.

The system will only estimate a service agreement's consumption if: 1) the bill cycle schedule allows estimation, and 2) the service agreement allows estimation.

Freeze and Complete Turn on this switch if the system should freeze and complete any bill that is created without errors. If this switch is turned off, all bills created by the billing process will be left in the unfinished state. You would only turn this switch off if you want to verify an entire bill run prior to freezing it (e.g., if you are introducing a new version of a rate). If you turn this off, you will need to return to this page after verifying a bill run and turn it back on for the customers to receive bills. When the system next runs, it deletes all unfrozen bills and recreates them as per the instructions on the bill cycle schedule.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BILL_CYC](#).

The batch bill creation process uses this schedule to define the bill cycles for which it should create bills.

NOTE:

Don't forget. After you set up the bill cycles that correspond with service cycles, go to [Setting Up Service Cycle Schedules](#) and update the service cycles accordingly.

Setting Up Service Route Types

Every route within a service cycle references a route type. The route type controls:

- The batch process used to download and upload a route's meter reads.
- Whether a route is downloaded as part of the service cycle schedule download process.

Open **Admin > Consumption > Service Route Type > Add** to define your route types.

Description of Page

The following fields display:

Route Type The unique identifier of the route type.

Description The description of the route type.

Batch Control This defines the batch process used to download the meter reads for routes of this type. If you have multiple meter reading device formats you will have multiple batch processes - one for each format.

Process Routes In Cycle Turn this switch on if routes of this type have their meter reads downloaded when their service cycle is downloaded. This switch will be on for most types of routes. A classic example of a type of route where this switch is off would be a route containing translators (a translator sends in reads when it is polled).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MR_RTE_TYPE](#).

Setting Up Service Cycles And Routes

When you set up a metered service point you must define its service cycle, route and sequence within the route. To define a service cycle and its routes, open **Admin > Service Cycle > Add**.

NOTE:

Every service cycle has a calendar that defines when the service points in the cycle are read. For more information about this calendar, see [Setting Up Service Cycle Schedules](#).

Description of Page

Enter a unique **Read Cycle** and **Description** for the service cycle.

Enter the **Bill Cycle** to populate on accounts with service points in this service cycle. The system updates an account's bill cycle when a service agreement is activated (assuming the account's bill cycle is not protected). Refer to [How Does An Account Get Its Bill Cycle?](#) for more information.

If the service cycle is associated with a **Service Provider** (rather than your own company's), define the owner of the cycle. Refer to [MDMAs And Service Cycles](#) for more information about service providers and their service cycles.

Use the **Service Routes** collection to define the routes within a given service cycle. The following fields display for each service route:

Service Route The unique identifier of the route within the cycle (you can use the same route ID in many cycles).

Description The route's description.

Route Type The type of route. This controls if and how the route's meter reads are downloaded. Refer to [Setting Up Service Route Types](#) for more information.

Characteristics

The **Characteristics** collection contains information that describes miscellaneous information about the service route.

The following fields display:

Effective Date Indicate the effective date for this characteristic.

Characteristic Type Controls the order in which characteristics of the same type are displayed.

Characteristic Value Indicate the value of the characteristic.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_MR_CYC](#) and [CI_MR_RTE](#).

Setting Up Service Cycle Schedules

The service cycle schedule defines when the service points in a given cycle are scheduled to be read. Open **Admin > Consumption > Service Schedule > Search** to maintain this information.

FASTPATH:

Refer to [Designing Cycles for Metered Services](#) for more information about service cycle. Refer to [The Relationship Between Metered Service Cycles and Bill Cycles](#) for more information about how service cycles are linked to bill cycles.

Description of Page

When you want to add a new service cycle schedule, you must specify the following information:

Service Cycle Enter the service cycle ID of the cycle whose routes will be downloaded.

Scheduled Selection Date Specify the date on which the system is meant to download information about the cycle's meters. This date should be a day or two before the scheduled work date.

Define the date the meters in the route are scheduled to be read using **Scheduled Work Date**. This date is extremely important as billing uses it when it looks for meter reads for service points in this cycle. If billing can't find a reading, consumption will be estimated as of this date (assuming the service agreement allows estimation).

Click **Pre-Generate Routes** if you need to finesse the cycle's routes on this Scheduled Selection Date (e.g., because some of the routes shouldn't be downloaded on a given date because the system is meant to estimate consumption). If you don't click this button, the system will create the routes on the scheduled selection date. It does this by creating a route extract for each downloadable route within the cycle (as defined by the route type). If you click this button, the system shows all routes within the cycle in the grid below. You must then define how the system is supposed to download each route on the scheduled selection date.

Click **Delete Routes** if you have pre-generated the routes and you want to remove them and allow the system to create the routes on the scheduled selection date.

Use the **Service Schedule Routes** to define how individual routes within the cycle should be handled during the download. The following fields are required for each schedule read date:

Service Route The unique ID of the service route. The route's route type is displayed adjacent.

Schedule Type This defines if and how the system is supposed to download the route's meter reads. Valid values are: Cust Read, Download, Estimate. Only those routes defined as Download will have meter reads downloaded. The other values are used to document why the route won't be downloaded.

Schedule Status This defines the download status of the route. Valid values are Pending and Complete. This value is only displayed for routes with a schedule type of Download.

Where Used

This information is used by the meter read download process to determine which meter reads to download.

This information is used by the billing process to determine the date on which it expects to find a read. If it cannot find a read on or around this date, and it's OK to estimate consumption, consumption will be estimated as of the scheduled work date. The system uses the service agreement's rate schedule's frequency to determine the period of time around the scheduled work date in which it looks for a read. Refer to [Defining Frequency Codes](#) for more information.

Setting Up Bill Periods

Some SA types reference a bill period. The bill period defines when the service agreement's bill segments are produced and the respective end date of each bill segment.

FASTPATH:

Refer to [Designing Bill Periods](#) for more information.

To define a bill period and the bill period schedule, open **Admin > Bill Period > Add**.

Description of Page

Enter a unique **Bill Period** and a **Description** for every bill period.

Use the **Bill Period Schedules** collection when the system should create bill segments for service agreements that use a given bill period. It also defines the end date of each respective bill segment. The following fields are required:

Bill Date Specify the earliest date on which the system is allowed to create a bill segment for service agreements using this bill period.

Bill Seg End Date Specify the end date of the bill segment. For future bills, this will be after the bill date. For retro bills, this will be before the bill date.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_BILL_PERIOD](#).

This information is used by the bill segment creation process to determine the end date of service agreements that use a bill period.

Defining Statement Cycles

If you have persons set up in the system to receive statements with financial information, you will need to assign them to a statement cycle and define a schedule for the statement cycle.

FASTPATH:

Refer to [The Big Picture of Complex Statements](#) for more information about statements.

The Big Picture Of Statement Cycles

A statement cycle has a similar purpose to that of a bill cycle. It controls when statements will be produced.

The Cyclical Statement Process

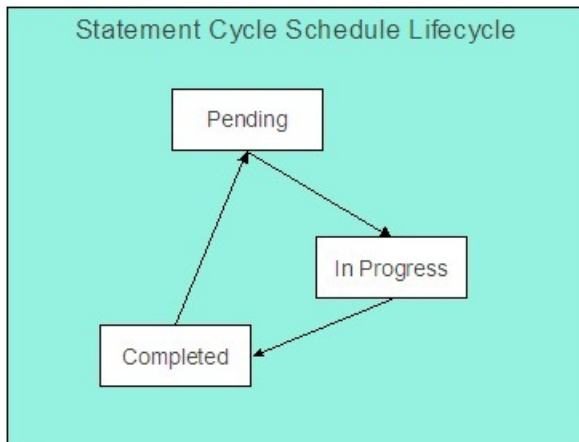
Persons who wish to receive statements will work with you to determine how often these statements should be produced. Some persons may want a monthly statement, some a quarterly and some annually. For each unique schedule that is designed for your various statement persons, you will set up a Statement Cycle and its schedule.

Designing Your Statement Cycles

The number of statement cycles is determined by a combination of the frequency that you will send statements to the statement persons and how many statement cycles you wish to manage within the same frequency.

So, for example, for all the statement persons who wish to receive a monthly statement, will you create only one monthly Statement Cycle so that all monthly statements are produced the same day? Or will you have several monthly statement cycles scheduled throughout the month? The answer will depend on the volume of statements being produced and on how you want to manage the statement production.

Lifecycle of a Statement Cycle Schedule



Pending The statement cycle schedule is added in this state. The Create Statements background process find records in this state to process on the appropriate date.

FASTPATH:

Refer to [Create Statements Background Process](#) for more information.

In Progress Records in this state are currently being processed by the Create Statements background process.

Completed Records in this state have already been processed by the Create Statements background process. If a problem occurs with the Create Statements background process and it needs to be rerun, simply change the status back to pending and rerun the process.

Setting Up Statement Cycles

A Statement Construct references a statement cycle. The statement cycle defines when the statement person will receive statements with financial information related to the accounts and service agreements linked to the statement construct. To define a statement cycle and its statement cycle schedule, open **Admin > Billing > Statement Cycle > Add**.

Description of Page

Enter a unique **Statement Cycle** and **Description** for every statement cycle.

Use the **Statement Cycle Schedule** collection to define when statements are produced for the persons with statement construct records in the given statement cycle. The following fields are required for each instance:

Processing Date Specify the date on which the system should create statements for persons with statement construct records in the cycle.

Status Indicates the status of the cycle schedule. Refer to [Lifecycle of a Statement Cycle Schedule](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_STM_CYC](#).

The batch statement creation process uses this schedule to determine which statement cycles for it should create statements for the statement construct records. Refer to [Create Statements Background Process](#) for more information.

Defining Service Agreement Type (SA Types)

Every service agreement must reference a SA type. The SA type defines what you sell, how much you sell it for, to whom you sell it, how overdue debt is collected, and how sales will be booked in your general ledger.

NOTE:

Perfect foresight. In a perfect world, the other control tables would have been set up with perfect foresight of setting up your SA types. In reality, setting up your SA types may invalidate some of your earlier decisions. Don't feel bad if this happens, some amount of iteration is natural.

Designing SA Types For Service Agreements With Service Points

Designing your SA types is an iterative process. To minimize some iterations, we recommend using the steps outlined in this section to complete the following table. When the table is complete, you're ready to set up your SA types.

Division/SA Type	Service Type
------------------	--------------

The topics in this section provide guidelines describing how to fill in this table for SA types associated with service agreements that charge for service point-oriented services.

CIS Division Segmentation

A CIS division is typically associated with a jurisdiction. The definition of a jurisdiction is a geographic-oriented entity with unique business rules. For example, if you conduct business in California and Nevada, and each state has different collection rules, you will need a separate jurisdiction for each state. You must set up a CIS division for each jurisdiction in which you conduct business.

If we assume that you are located in a single jurisdiction - say California - we will need a single CIS division for all of our SA types.

CIS Division/SA Type

CA

Service Segmentation

At a minimum, you will have one SA type for every different type of utility service offered by your organization. If we assume you sell electricity, gas, water, wastewater and cable; your SA Types will be as follows:

CIS Division/SA Type	Service Type
CA/G	Gas service
CA/W	Water service
CA/E	Electric service
CA/WW	Waste water service
CA/CABLE	Cable

NOTE:

Non Utility Services. Earlier in this manual, service types are discussed in respect of meters and items. However, you may require additional service types if you have non-utility services. In the above table, we have only shown utility oriented services and their respective CIS Divisions. Later in this section, we will encounter a few more service types. Refer to [Setting Up Service Types](#) for more information.

Receivable Segmentation

Many organizations segregate their receivable balances in the general ledger. For example, the receivable amount associated with gas and water service may be maintained in separate GL accounts.

If your organization does this, you will probably have at least one SA type for each such receivable account because each SA type references a distribution code that typically contains the receivable account.

- The word *probably* is underlined because this is a rule of thumb. There are situations where the number of receivable accounts isn't directly related to the number of SA types. This happens when an organization maintains very detailed receivable accounts in the general ledger and maintaining a one-to-one relationship between SA types and distribution codes would lead to a massive proliferation of SA types (and you don't want this!). If your organization maintains very detailed receivable accounts, please speak to your implementers, they should be able to introduce a small customization to generate the appropriate receivable account rather than extract it from the distribution code.
- The word *typically* is underlined because there are several SA types that don't book to a receivable account when bill segments are generated. For example, company usage and charitable contributions. Refer to [Company Usage Segmentation](#) and [Charitable Contribution Segmentation](#) for examples of SA types that don't book to receivable accounts.

We'll simplify our example and assume your organization has one receivable account for all types of utility service. Given this, we won't need additional SA types to support receivable segmentation:

CIS Division/ SA Type	Service Type	Distribution Code
CA/G	Gas service	A/R-UTIL

CA/W	Water service	A/R-UTIL
CA/E	Electric service	A/R-UTIL
CA/WW	Waste water service	A/R-UTIL
CA/CABLE	Cable service	A/R-UTIL

Revenue Segmentation

Look at your rates and determine which rates can be used by each SA type. The following table shows the sample rates that can be used for each service:

CIS Division/ SA Type	Service Type	Distribution Code	Rates
CA/G	Gas service	A/R-UTIL	GALL-1
CA/W	Water service	A/R-UTIL	WALL-1
CA/E	Electric service	A/R-UTIL	ERES-1, ERES-2, ECOM-1, EIND-1, ELAMP-1
CA/WW	Waste water service	A/R-UTIL	WWALL-1
CA/CABLE	Cable service	A/R-UTIL	CABLE

Now, look at the rates' calculation rule GL Distribution window. You're looking for calculation rules whose GL distribution is affected by revenue class. If there are no revenue classes referenced on the calculation rules, this means that the revenue associated with the rate will be booked to a single GL account regardless of the type of customer. If you see revenue classes, this means that the revenue account associated with the calculation rule(s) differs depending on the SA type's revenue class. If revenue classes are used in the rates, you must create a different SA Type for every revenue class.

Let's assume the following:

- The gas rate (GALL-1) references the RESIDENTIAL, COMMERCIAL and INDUSTRIAL revenue classes in order to differentiate revenue based on the type of customer.
- None of the other rates differentiate revenue based on customer class.

Our SA types will now look as follows:

CIS Division/ SA Type	Service Type	Distribution Code	Revenue Class	Rates
CA/G-RES	Gas service	A/R-UTIL	R - residential	GALL-1
CA/G-COM	Gas service	A/R-UTIL	C - commercial	GALL-1
CA/G-IND	Gas service	A/R-UTIL	I - industrial	GALL-1
CA/W	Water service	A/R-UTIL	N/A	WALL-1
CA/E	Electric service	A/R-UTIL	N/A	ERES-1, ERES-2, ECOM-1, EIND-1, ELAMP-1
CA/WW	Waste water service	A/R-UTIL	N/A	WWALL-1
CA/CABLE	Cable service	A/R-UTIL	N/A	CABLE

Notice that we created new SA types for gas in order to specify the respective revenue class. We didn't do this for the other services because it isn't necessary. However, you should feel free to do this if it feels right or if you need it for reporting purposes. For example, if you want to report on all *residential* service agreements, but you differentiate only gas by residential versus commercial, you'll be in trouble.

Rate Segmentation

Every SA Type whose service agreements have their charges calculated with a rate must have one or more rates linked to it. These define which rates can be linked to the SA type's service agreements. If different service agreements have different valid rate combinations, you will need to split the SA types accordingly.

For example, look at the electric rates. If we assume:

- the RES rates can only be used for residential customer,
- the IND rates can only be used for industrial customers,
- the COM rates can only be used for commercial customers,
- the LAMP rates can only be used for lamp customers

Then you might want to set up a new SA type to indicate such. The advantage of doing this is that you get more control over which rates can be used on a given SA type. The disadvantage is that you proliferate SA types. Unfortunately, it's really a question of taste.

CIS Division/ SA Type	Service Type	Distribution Code	Revenue Class	Rates
CA/G-RES	Gas service	A/R-UTIL	R - residential	GALL-1
CA/G-COM	Gas service	A/R-UTIL	C - commercial	GALL-1
CA/G-IND	Gas service	A/R-UTIL	I - industrial	GALL-1
CA/W	Water service	A/R-UTIL	N/A	WALL-1
CA/E-RES	Electric service	A/R-UTIL	N/A	ERES-1, ERES-2
CA/E-COM	Electric service	A/R-UTIL	N/A	ECOM-1
CA/E-IND	Electric service	A/R-UTIL	N/A	EIND-1
CA/LAMP	Electric service	A/R-UTIL	N/A	ELAMP-1
CA/WW	Waste water service	A/R-UTIL	N/A	WWALL-1
CA/CABLE	Cable service	A/R-UTIL	N/A	CABLE

WARNING:

Don't be too specific in your analysis in respect of rate segmentation because you could end up with a separate SA type for every rate (and you don't want this). We understand this is a very subjective warning, but we recommend that you start out with broad bands of rates that can be used on a SA type and narrow it down if you end up unhappy with the results. For example, you don't have to set up a separate SA type for low-income residential gas customers just because they have a special rate. Rather, you can leave your SA types as they are and treat low-income gas customers as a subset of your residential gas customers. **For more information**, refer to [Setting Up Start Options](#).

NOTE:

Rate override. If a service agreement of this type may be linked to a terms of service record, you must also consider whether or not the [rate schedule could be overridden](#) by a template SA linked to the terms of service record. Refer to [SA Type - Rate](#) for information about the possible values for this field.

Service Point (SP) Type Segmentation

Every SA Type whose service agreements exist to bill for service point-oriented service has one or more SP types. These define which type of service points can be linked to the SA type's service agreements. If different service agreements have different valid SP types, you will need to split the SA types accordingly.

For each service point-oriented SA type, determine if there are any restrictions in respect of the types of service points that can use the SA type's rates. For example, if we assume that only commercial SP types can be used by commercial customers, industrial SP types by industrial customers, residential SP types by residential customers, your SA types will be:

CIS Division/ SA Type	Service Type	Distribution Code	Revenue Class	Rates	SP Type
CA/G-RES	Gas service	A/R-UTIL	R - residential	GALL-1	G-RES
CA/G-COM	Gas service	A/R-UTIL	C - commercial	GALL-1	G-COM
CA/G-IND	Gas service	A/R-UTIL	I - industrial	GALL-1	G-IND
CA/W-RES	Water service	A/R-UTIL	N/A	WALL-1	W-RES
CA/W-COM	Water service	A/R-UTIL	N/A	WALL-1	W-COM
CA/W-IND	Water service	A/R-UTIL	N/A	WALL-1	W-IND
CA/E-RES	Electric service	A/R-UTIL	N/A	ERES-1, ERES-2	E-RES
CA/E-COM	Electric service	A/R-UTIL	N/A	ECOM-1	E-COM
CA/E-IND	Electric service	A/R-UTIL	N/A	EIND-1	E-IND
CA/WW-RES	Wastewater service	A/R-UTIL	N/A	WWALL-1	W-RES, WW-RES
CA/WW-COM	Wastewater service	A/R-UTIL	N/A	WWALL-1	W-COM, WW-COM
CA/WW-IND	Wastewater service	A/R-UTIL	N/A	WWALL-1	W-IND, WW-IND
CA/CABLE	Cable service	A/R-UTIL	N/A	CABLE	CABLE

Notice the wastewater SA types reference both water and waste water service points. This is intentional as wastewater service uses the consumption from the water service to calculate some part of the wastewater charge.

Company Usage Segmentation

Up to now, we've discussed SA types associated with service agreements linked to your customers. The system has also been designed to keep track of the expenses associated with your company's use of power. If you want the system to do this, you must create at least one SA type for each service consumed by your organization.

For example, if we assume your organization consumes electric, gas, and water service; your SA types will now be as follows:

CIS Division/ SA Type	Service Type	Distribution Code	Revenue Class	Rates	SP Type	Bill Seg Type
CA/G-RES	Gas service	A/R-UTIL	R - residential	GALL-1	G-RES	SP-RATED

CA/G-COM	Gas service	A/R-UTIL	C - commercial	GALL-1	G-COM	SP-RATED
CA/G-IND	Gas service	A/R-UTIL	I - industrial	GALL-1	G-IND	SP-RATED
CA/W-RES	Water service	A/R-UTIL	N/A	WALL-1	W-RES	SP-RATED
CA/W-COM	Water service	A/R-UTIL	N/A	WALL-1	W-COM	SP-RATED
CA/W-IND	Water service	A/R-UTIL	N/A	WALL-1	W-IND	SP-RATED
CA/E-RES	Electric service	A/R-UTIL	N/A	ERES-1, ERES-2	E-RES	SP-RATED
CA/E-COM	Electric service	A/R-UTIL	N/A	ECOM-1	E-COM	SP-RATED
CA/E-IND	Electric service	A/R-UTIL	N/A	EIND-1	E-IND	SP-RATED
CA/WW-RES	Wastewater service	A/R-UTIL	N/A	WWALL-1	W-RES, WW-RES	SP-RATED
CA/WW-COM	Wastewater service	A/R-UTIL	N/A	WWALL-1	W-COM, WW-COM	SP-RATED
CA/WW-IND	Wastewater service	A/R-UTIL	N/A	WWALL-1	W-IND, WW-IND	SP-RATED
CA/CABLE	Cable service	A/R-UTIL	N/A	CABLE	CABLE	SP-RATED
CA/E-COY	Electric service	EXP-COMP	N/A	E CO USE	E-CO USE	COMPUSAG
CA/G-COY	Gas service	EXP-COMP	N/A	G CO USE	G-CO USE	COMPUSAG
CA/W-COY	Water service	EXP-COMP	N/A	W CO USE	W-CO USE	COMPUSAG

Notice the three company usage SA types do not reference an A/R account as their distribution code. This is because when bill segments are created for these types of service agreements, the system must debit an expense account (or contra-revenue account) rather than a receivable account.

Also notice we introduced a new column - Bill Segment Type. Notice that the customer-oriented SA types use the SP-RATED bill segment type and the company usage SA types use the COMPUSAG bill segment type. Different bill segment types are necessary because company usage SA types use a different algorithm to calculate their bill segment's financial transaction algorithm because they don't affect either payoff or current balance.

FASTPATH:

For more information, refer to [Designing and Defining Bill Segment Types](#).

Debt Class Segmentation

Every SA Type has a debt class. The debt class is used to categorize a service agreement's debt for the purpose of credit and collections (C&C) analysis. If a given SA Type has different categories of debt from C&C's perspective, you will have to split the SA Type.

FASTPATH:

For more information about debt class, refer to [Designing Your Collection Procedures](#).

If we assume that your residential services are regulated and your commercial and industrial services are deregulated, we won't have to introduce additional SA types.

CIS Division/ SA Type	Distribution Code	Revenue Class	Rates	SP Type	Bill Seg Type	Debt Class
CA/G-RES	A/R-UTIL	R - residential	GALL-1	G-RES	SP-RATED	REGU

CA/G-COM	A/R-UTIL	C - commercial	GALL-1	G-COM	SP-RATED	UNRE
CA/G-IND	A/R-UTIL	I - industrial	GALL-1	G-IND	SP-RATED	UNRE
CA/W-RES	A/R-UTIL	N/A	WALL-1	W-RES	SP-RATED	REGU
CA/W-COM	A/R-UTIL	N/A	WALL-1	W-COM	SP-RATED	UNRE
CA/W-IND	A/R-UTIL	N/A	WALL-1	W-IND	SP-RATED	UNRE
CA/E-RES	A/R-UTIL	N/A	ERES-1, ERES-2	E-RES	SP-RATED	REGU
CA/E-COM	A/R-UTIL	N/A	ECOM-1	E-COM	SP-RATED	UNRE
CA/E-IND	A/R-UTIL	N/A	EIND-1	E-IND	SP-RATED	UNRE
CA/WW-RES	A/R-UTIL	N/A	WWALL-1	W-RES, WW-RES	SP-RATED	REGU
CA/WW-COM	A/R-UTIL	N/A	WWALL-1	W-COM, WW-COM	SP-RATED	UNRE
CA/WW-IND	A/R-UTIL	N/A	WWALL-1	W-IND, WW-IND	SP-RATED	UNRE
CA/CABLE	A/R-UTIL	N/A	CABLE	CABLE	SP-RATED	UNRE
CA/E-COY	EXP-COMP	N/A	E CO USE	E-CO USE	COMPUSAG	No debt
CA/G-COY	EXP-COMP	N/A	G CO USE	G-CO USE	COMPUSAG	No debt
CA/W-COY	EXP-COMP	N/A	W CO USE	W-CO USE	COMPUSAG	No debt

Notice the three company usage SA types do not have a debt class. This is because their bill segment type's FT algorithm doesn't cause debt to be created and therefore there is no reason to have a debt class. However, you'll need to create a "dummy" debt class - call it N/A - for these SA types because every SA type must reference a debt class.

Budget Billing Segmentation

Many utilities offer their customers levelized payment plans to smooth out the seasonal bill variations. We call this levelized amount the *budget amount*.

FASTPATH:

Refer to [Budget Billing](#) for more information about budgets in general. Refer to [Billing - Current Balance versus Payoff Balance](#) for an example of budget billing accounting.

If we assume that you only allow budget billing on the electric and gas residential services, then you'll need to update your CA/G-RES and CA/E-RES SA types:

CIS Division/ SA Type	Dist Code	Rev Class	Rates	SP Type	Bill Seg Type	Debt Class	Recurring Charge Control Info
CA/G-RES	A/R-UTIL	R	GALL-1	G-RES	SP-RATED	REGU	Amount to bill is Not Allowed Amount is Optional Frequency is Monthly Recurring Amount Label

							is Budget Amount:
CA/G-COM	A/R-UTIL	C	GALL-1	G-COM	SP-RATED	UNRE	
CA/G-IND	A/R-UTIL	I	GALL-1	G-IND	SP-RATED	UNRE	
CA/W-RES	A/R-UTIL	N/A	WALL-1	W-RES	SP-RATED	REGU	
CA/W-COM	A/R-UTIL	N/A	WALL-1	W-COM	SP-RATED	UNRE	
CA/W-IND	A/R-UTIL	N/A	WALL-1	W-IND	SP-RATED	UNRE	
CA/E-RES	A/R-UTIL	N/A	ERES-1, ERES-2	E-RES	SP-RATED	REGU	Amount to bill is Not Allowed Amount is Optional Frequency is Monthly Recurring Amount Label is Budget Amount:
CA/E-COM	A/R-UTIL	N/A	ECOM-1	E-COM	SP-RATED	UNRE	
CA/E-IND	A/R-UTIL	N/A	EIND-1	E-IND	SP-RATED	UNRE	
CA/WW-RES	A/R-UTIL	N/A	WWALL-1	W-RES, WW-RES	SP-RATED	REGU	
CA/WW-COM	A/R-UTIL	N/A	WWALL-1	W-COM, WW-COM	SP-RATED	UNRE	
CA/WW-IND	A/R-UTIL	N/A	WWALL-1	W-IND, WW-IND	SP-RATED	UNRE	
CA/CABLE	A/R-UTIL	N/A	CABLE	CABLE	SP-RATED	UNRE	
CA/E-COY	EXP-COMP	N/A	E CO USE	E-CO USE	COMPUSAG	No debt	
CA/G-COY	EXP-COMP	N/A	G CO USE	G-CO USE	COMPUSAG	No debt	
CA/W-COY	EXP-COMP	N/A	W CO USE	W-CO USE	COMPUSAG	No debt	

Notice the following:

- We updated the two SA types to allow recurring charge information.
- The Recurring Charge Amount is Optional. Why? Because those customers who aren't on a budget plan won't have a recurring charge amount. Those customers on a budget will have a recurring charge amount.

NOTE:

Turn on Eligible for Budget. Besides indicating that these SA types use recurring charges, you also need to turn on the Eligible for Budget switch on the SA type to indicate that this type of SA participates in budget processing.

Override Budget Eligibility. You may plug in an override budget eligibility algorithm on an SA type that is configured to be Eligible for Budget if certain service agreements of this type are not eligible.

FASTPATH:

Refer to the Description of Page under [SA Type - Billing](#) for the definition of the recurring charge attributes.

Designing SA Types For SAs Without Service Points

The topics in this section provide guidelines describing how to design the SA types associated with your service agreements that don't have service points.

Overpayment Segmentation

When a customer pays more than they owe, you must decide what to do with the excess money. The following points describe two possibilities:

- You could create a new service agreement to hold the excess (let's call it an overpayment SA). The credit would be transferred from this service agreement to the billable service agreements when the next bill is completed. This means that all billable service agreements have the same opportunity to receive the overpayment when they are billed in the future.
- You could amalgamate the excess payment on one of the existing, billable service agreements. For example, if a customer has both electric and gas service, the excess funds could be kept on either the gas or the electric SA. This would result in the following:
 - The service agreements that do NOT receive the overpayment will have debt when they are next billed.
 - The service agreement that receives the overpayment could have its future debt offset by the overpayment (meaning that it could have a credit balance until the service agreement's future bill segments offset the overpayment amount).
- The above situation is not desirable unless the customer intentionally overpaid one service agreement. The first method (keeping the overpayment on a separate service agreement) obviates this potential problem. Obviously, if your organization sells a single service (and therefore your customers have a single service agreement) you would choose the second method.

You control which method is used by plugging in the appropriate Overpayment Distribution algorithm on each [Customer Class](#) (i.e., you can choose a different method for different customer classes). If you choose to hold overpayments on a separate SA, then you must set up an SA Type as described in the following table:

CIS Division/ SA Type	Service Type	Distrib. Code	Eligible for Billing	Debt Class	Pay Seg Type	Do Not Overpay	One-time
CA/OVERPAY	Other	A/P - OVER	Not billed	N/A	Normal	No	Yes

Notice the following about the new overpayment SA type:

- It has an interesting distribution code. This is because when a payment segment is created for this type of service agreements, the system must credit a liability (an overpayment is a liability).
- It's important to indicate that the overpayment SA is a one-time service agreement. Why? Because this means that the system will automatically close the SA when its balance falls to zero (i.e., when all of the overpayment has been used to satisfy future bills).
- A bill segment type is not needed because the system never creates bill segments for such service agreements (they exist only to hold excess credits).
- You may also want to turn on the alert message
- You must plug-in a bill completion algorithm on this SA type. This bill completion algorithm will transfer the credit balance to the account's other service agreements when the bill is completed. Refer to [The Credit Transfer Algorithm](#) for more information about this algorithm.

- You must also reference this overpayment SA type as the parameter value on your overpayment algorithm (this algorithm is plugged in on the desired customer classes). Refer to [Overpayment Algorithm](#) for more information about this algorithm.

NOTE:

If overpayment means charitable contribution. Some organizations sponsor a program that works as follows - if a customer overpays a bill by a given amount (say \$5), this amount is assumed to be a charitable contribution. If you have this requirement, you will create another SA type to hold a customer's charitable contributions. This SA type will look similar to the one described below (see [Charitable Contribution Segmentation](#)) except it is not billable. The funds will be credited to this service agreement by creating a new overpayment algorithm that is similar to the base package [Overpayment Algorithm](#). This new algorithm will be very similar to the existing algorithm. The main difference will be that it will have to check if the overpayment amount is an exact value (say \$5). If so, it will create a payment segment for the charitable contribution SA type; otherwise it will create a payment segment for the overpayment SA.

Write Off Segmentation

When you write off non-collectable debt, you transfer the receivable from a "normal" service agreement onto one or more write-off service agreements. When the debt is transferred to a write-off service agreement, the distribution code on the "normal" service agreement is credited (typically an A/R GL account), and the distribution code on the write-off service agreement is debited.

You will almost always need a write-off service agreement whose distribution code is the write-off expense. However, you probably don't book all of the write-off amount to a write-off expense account. Why? Because the debt that you're writing off typically contains both revenue and liabilities. At write-off time, you want to book the written off revenue to a write-off expense account and you want to reduce the liabilities (you don't owe the liability if you don't get paid). This means you'll need another SA type for the liabilities. Refer to [The Ramifications of Write Offs in the General Ledger](#) for a complete explanation.

The following table contains the minimum number of SA Types that you'll need to hold your write-offs.

CIS Division/ SA Type	Service Type	Distrib. Code	Bill Seg Type	Debt Class	Pay Seg Type	Do Not Overpay
CAWO-STD	Other	EXP-W/O	Not billed	WO	Normal	Yes
CAWO-LIA	Other	LIA-General	Not billed	WO	Normal	Yes

Notice the following about the new write-off SA types:

- They have interesting distribution codes. This is because when debt is transferred to these types of service agreements, the system must debit either an expense account (i.e., write-off expense) or a liability account. It's important to note that in [The Ramifications of Write Offs in the General Ledger](#) we explain how this liability account may be overwritten with the liability account that was originally booked.
- Neither needs a bill segment type because the system never creates bill segments for such service agreements (they exist only to hold uncollectable debt)
- Even though the debt is not collectable, it still has a debt class. Why? Because the system shows a customer's debt on many inquiries by debt class and it's important to show write-off debt on these queries.
- The combination of Payment Segment Type and Do Not Overpay are important. Refer to [The Ramifications of Write Offs in the General Ledger](#) for a complete explanation.

NOTE:

The adjustment type used to set the offending service agreement's current balance equal to its payoff balance is defined on each write-offable SA type. The adjustment type used to transfer the delinquent debt to the write-off service agreement is defined on the write-off SA type.

An Alternative. If you have a limited number of liability accounts, you may decide to have a separate write-off service agreement for each liability account. Doing this would proliferate the number of service agreements created at write-off time. However, it would simplify the remittance of payment to the taxing authority if the reversed liability is ever paid.

Connection Charge Segmentation

If you levy connection charges, you have two options:

- You can create a SA type that exists purely to handle connect charge debt. After doing this, you'd create a start option for this SA type that causes an adjustment to be levied as part of the start service process. This adjustment would contain your standard connection charge. This approach would be used by a utility that had multiple services (e.g., a combined electric, gas, water utility) that only levies a single connection charge regardless of the number of services started. If you use this approach, make sure to indicate the SA type is non billable.
- You can levy a start adjustment on one of your existing SA types (e.g., CA/E-RES). The easiest way to do this is with a start option. On the start option you'd indicate an adjustment to be levied as part of the start service process. This adjustment would contain your standard connection charge. This approach would be used by a utility that had a single service offering (e.g., an electric-only utility). Refer to [Setting Up Start Options](#) for more information.

In the table below, we show what would be necessary if you want to have a separate service agreement for the connection charge.

CIS Division/ SA Type	Service Type	Distrib. Code	Bill Seg Type	Debt Class
CA/CONNECT	Other	A/R-UTIL	Not billed	REGU

Notice the following about the new connection charge SA type:

- It has a normal receivable distribution code.
- It doesn't need a bill segment type because the system never creates bill segments for such service agreements (its charged via an adjustment).
- The debt class is interesting - REGU (for regulated). We are intentionally linking the connection charge debt to the same debt class as the regulated debt from which it originates. This way, the C&C process will consider the connection charge debt the same as regulated debt and therefore perform the regulated collection (which results in the severance of all regulated service agreements).

Charitable Contribution Segmentation

If your organization accepts charitable contributions made by your customers, you must create a SA type to hold these contributions.

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/CHARITY	Other	A/P-CHAR	CHAR	RECUR	Amount to bill is Not Allowed Amount is Required Frequency is Monthly

Recurring Amount
Label is Contrib.
Amount

Notice the following about the new charitable contribution SA type:

- It has an interesting distribution code. This is because when a payment is distributed to these types of service agreements, the system must credit a payable account (i.e., charitable contribution payable) rather than a receivable account. Note well, we have assumed a receivable is not incurred when the bill segment for the charitable contribution is created.
- It uses an interesting bill segment type - RECUR. This bill segment type was set up to create recurring charges that don't automatically stop at some point in time.
- The debt class is interesting - CHAR (for charity). This is done so that past due charitable contribution debt is treated separately from other types of debt.
- The recurring charge control information is set up as defined.

FASTPATH:

Refer to the Description of Page under [SA Type - Billing](#) for the definition of the recurring charge attributes. Refer to [Start Option Considerations For SA Types That Use Recurring Charges](#) for how you can use start options to automatically populate a service agreement's recurring charge fields with appropriate values when service is started.

WARNING:

It's important that you assign the charitable contribution SA type with a payment segment type that only affect current balance (as opposed to affecting current AND payoff balance). This is because there is no receivable recognized when the contribution is billed and therefore there is no payoff balance to relieve when it's paid.

Payment Arrangement Segmentation

If your organization allows customers to payoff outstanding debt using payment arrangements (e.g., current bill plus \$X), you will need a new SA type for every debt class that can have a payment arrangement. If we assume you can have payment arrangements for both regulated and unregulated debt, then you'll need at least two more SA types (you may have more SA types if you need to segregate the payment arrangement receivable amount by utility type (or some other type)).

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/PA-REGU	Other	A/R-ARRG	REGU	RECUR-AS	Amount to bill is Not Allowed Amount is Required Frequency is Monthly Recurring Amount Label is Arrange Amount
CA/PA-UNRE	Other	A/R-ARRG	UNRE	RECUR-AS	Amount to bill is Not Allowed Amount is Required Frequency is Monthly

Recurring Amount
Label is Arrange
Amount

Notice the following about the new payment arrangement SA types:

- They have an interesting distribution code. This is because when funds are transferred to these types of service agreements, the system must debit a receivable (i.e., payment arrangement receivable).
- They use an interesting bill segment type - RECUR-AS. This bill segment type was set up to create recurring charges that stop when the customer no longer has a payoff balance.
- Each new SA type references the debt class whose debt it will pay off. We are intentionally linking the payment arrangement debt to the same debt class as the regulated debt from which it originates. This way, the C&C process will consider the arrangement debt as the same as regulated debt and therefore perform the regulated collection (which results in the severance of all regulated service agreements).
- The recurring charge control information is set up as defined.

FASTPATH:

Refer to the Description of Page under [SA Type - Billing](#) for the definition of the recurring charge attributes. Refer to [Start Option Considerations For SA Types That Use Recurring Charges](#) for how you can use Start Options to automatically populate a service agreement's recurring charge fields with appropriate values when service is started.

Merchandise Segmentation - Installment Billing

If your organization allows customers to purchase merchandise using an installment plan, you must create a SA type for this.

NOTE:

No installments. If the customer must pay for the merchandise in one lump amount, you'd create an SA type similar to the [Connection Charge Segmentation](#) example.

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/MERCH-I	Merch	A/R-MRCH	UNRE	RECUR-AS	Amount to bill is Not Allowed Amount is Required Frequency is Monthly Recurring Amount Label is Install Amount

Notice the following about the new merchandise SA type:

- It has a normal receivable distribution code.
- It uses an interesting bill segment type - RECUR-AS. This bill segment type was set up to create recurring charges that stop when the customer no longer has a payoff balance.
- The recurring charge control information is set up as defined.

FASTPATH:

Refer to the Description of Page under [SA Type - Billing](#) for the definition of the recurring charge attributes. Refer to [Start Option Considerations For SA Types That Use Recurring Charges](#) for how you can use start options to automatically populate a service agreement's recurring charge fields with appropriate values when service is started. Refer to [Start Option Considerations For SA Types That Use Initial Adjustments](#) for how you can use start options to automatically populate a service agreement's recurring charge fields with appropriate values when service is started.

Deposit Segmentation - Installment Billing

If your organization allows customers to pay deposits using an installment plan, you must create an SA type for this.

NOTE:

No installments. If the customer must pay for the deposit in one lump amount, you'd create an SA type similar to the [Connection Charge Segmentation](#) example. Just make sure the adjustment that's levied to charge for the deposit amount doesn't affect payoff balance (when you bill a deposit, the customer doesn't really owe anything because it's not a true receivable from an accountant's perspective).

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/DEP-I	Other	A/P-DEPO	DEP	RECURATB	Amount to bill is Required Amount is Required Frequency is Monthly Recurring Amount Label is Install Amount

Notice the following about the new deposit SA type:

- It has an interesting distribution code. This is because when a payment is distributed to these types of service agreements, the system must credit a payable account (i.e., deposit payable) rather than a receivable account. Note well, we have assumed a receivable is not incurred when the bill segment for the deposit is created.
 - It uses an interesting bill segment type - RECURATB. This bill segment type was set up to create recurring charges that stop when the system has billed the Total Amount to Bill.
 - The debt class is interesting - DEP (for deposit). This is done so that past due deposit "debt" is treated separately from other types of debt.
 - The recurring charge control information is set up as defined. Note well, the Amount to bill is Required.
-

FASTPATH:

Refer to the Description of Page under [SA Type - Billing](#) for the definition of the recurring charge attributes. Refer to [Start Option Considerations For SA Types That Use Recurring Charges](#) for how you can use start options to automatically populate a service agreement's recurring charge fields with appropriate values when service is started.

NOTE:

Bill messages on receipt of deposit in full. The base package includes a special FT Freeze algorithm that can be specified on deposit SA Types. It recognizes when a deposit has been paid in full, and creates a bill messages to inform the customer. Refer to algorithm DEP PIF MSG in [Algorithm Types](#) for more information.

Billable Charge Segmentation

You create a billable charge whenever a customer should be charged for a service that occurs outside the normal course of business. For example, you would create a billable charge to charge a contractor for the repair of a ruptured gas line. You can also use billable charges to "pass through" other bill ready charges generated outside the system, by another application, or by a 3rd party supplier.

A billable charge must reference a service agreement. This service agreement behaves just like any other service agreement:

- **Bill segments are created for the service agreement.** Whenever billing is performed for an account with billable charge service agreements, the system creates a bill segment for each service agreement with unbilled charges. If multiple unbilled charges exist for a given service agreement, only one bill segment will be created and it will contain details about all of the billable charges.
- **Payments are distributed to the service agreement.** Payments made by an account are distributed to its billable charge service agreements just like any other service agreement.
- **Overdue debt is monitored.** The credit and collections process monitors billable charge service agreements for overdue debt and responds accordingly when overdue debt is detected.

Therefore, you must set up at least one SA type to hold your billable charge debt. You may have multiple charges based on billing frequencies, A/R booking, debt monitoring, etc. It's really up to you.

The easiest way to determine how many billable charge SA types you'll need is to define every conceivable billable charge (which you should have done when you designed your [billable charge templates](#)). Then ask yourself if they have the same billing and payment behavior, if so, you'll have one SA type. If not, you'll need one SA type for each combination.

We will assume your billable charges are all used to levy unusual one-off charges that can be collected in the same way, therefore we'll need one SA type.

CIS Division/ SA Type	Service Type	Distrib Code	Debt Class	Bill Seg Type	Billable Charge Templates	Rate
CA/ONETIME	Other	A/R-UTIL	UNRE	BILLCHRG	TREETRIM DAMAGE	None
CA/PASSTHRU	Electricity	A/R-OTHER	UNRE	BILLCHRG	None	None
CA/ADDON	Electricity	A/R-OTHER	EXTERNAL	BILLCHRG	None	TAXES

Notice the following about the new one time SA type:

- It has a normal receivable distribution code.
- Its debt class is unregulated.
- It uses an interesting bill segment type - BILLCHRG. This bill segment type was set up to create bill segments using billable charges.
- It references the valid billable charge templates that can be used on this SA type.

NOTE:

One Time Charge. The ONE TIME example shown above implies this SA type exists to hold one-time charges. Because of this, you should turn on the One Time Charge switch on the SA type so that service agreement's of this type are automatically closed when final payment is received. You don't have to do this because a customer could have a

single billable charge service agreement that is perpetually active for pass through charges (i.e., it doesn't have a stop date). If you do this, the system will create a bill segment for this service agreement whenever it finds an unbilled billable charge linked to the service agreement.

Notice the following about the pass through SA Type:

- It doesn't use the normal distribution code or debt class. This is done so that the debt and receivable can be tracked separately. If these charges were being pass through from another system, you might want to track these financial values separately.
- It still uses the normal bill segment type - BILLCHRG. From a billing perspective, there is no difference between this and the one time SA Type.
- Templates are not relevant - these charges are not created on-line using templates, but are loaded via the [Billable Charge Upload Staging](#).

Notice the following about the add on charges SA Type:

- This is an example of bill-ready charges (similar to pass through) to which the system adds on other charges, for example, taxes.
- It still uses the normal bill segment type - BILLCHRG. From a billing perspective, there is no difference between this and the one time SA Type.
- It also uses a Rate. In this case, the bill creation algorithm (specified on the bill segment type) will take any billable charge lines and attach them to a bill. In addition, these billable charges will include billable charge service quantities (SQs). These service quantities will also be swept onto the bill segment, and the Rate (TAXES in this example) will be applied. In order for taxes to be calculated, the billable charge SQs must include the total taxable amount - the system is not able to apply the rate on top of the other billable charges. But, it can apply the tax rate to the SQs that are supplied.
- You can also use this technique to bill other rate-ready service quantities, like kWh, CCF, etc. This is a way to process rate-ready data for which you have a contract, but you do not know the meter (and therefore, cannot collect real meter reads).
- If the rate has pre-processing calculation groups, these will be applied as well.

FASTPATH:

For more information about billable charge templates, refer to [Setting Up Billable Charge Templates](#).

Over/Under Cash Drawer Segmentation

In order to balance a tender control that is out-of-balance, your organization must set up an account with a service agreement whose SA type references the over/under expense account. You will probably only have one service agreement that references this SA type, but you still must have it if you remit funds via a cash drawer.

FASTPATH:

For more information about over/under processing, refer to [How To Get An Unbalanced Tender Control In Balance \(Fixing Over/Under\)](#).

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type
CA/OVR UNDR	Other	EXP-OV/UND	N/A	Not billed

Notice the following about the new SA type:

- It has an interesting distribution code. This is because when a payment segment is applied to this type of service agreement, the system must debit an expense account for under amounts (and credit it for over amounts).
- It doesn't need a bill segment type because the system never creates bill segments for such service agreements (it only has over/under payment segments linked to it).
- It uses the N/A debt class because the credit and collections process should never consider debt associated with service agreements of this type (because it's not really debt).

Payment Upload Error Segmentation

If the payment upload process detects an invalid account on a payment upload record, it will create a payment for the suspense service agreement defined on the upload process' tender source (see [Setting Up Tender Sources](#)). You should create a special SA type for this service agreement.

FASTPATH:

For more information about the payment upload process, refer to [Phase 3 - Create Payment Events, Tenders, Payments, and Payment Segments](#).

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type
CA/SUSPENSE	Other	EXP-MISC	N/A	Not billed

Notice the following about the new SA type:

- It has an interesting distribution code. This code should probably be a suspense account. All payment segments that are created for this service agreement will eventually be transferred to a "real" service agreement and therefore this GL account's balance should be zero when no payments are in suspense.
- It doesn't need a bill segment type because the system never creates bill segments for such service agreements (it only has invalid account payment segments linked to it).
- It uses the N/A debt class because the credit and collections process should never consider debt associated with service agreements of this type (because it's not really debt).

CIAC Segmentation

If your company bills and refunds Contribution In Aid of Construction (CIAC) contracts, you must create one or more SA types. CIAC contracts are typically used to levy charges associated with line extensions. These types of service agreements are different from other service agreements because the initial amount charged is refunded to the original payee when new properties (or extensions) are added to the extension (or when the new properties are subsequently billed for service).

NOTE:

Billable charges are used for the original CIAC service agreement. CIAC SA types are always Billable Charge SA types because an operator must specify the exact amount to charge the contractor using a billable charge. Refer to [Billable Charge Segmentation](#) for more information.

The following points describe how CIAC processing is implemented in the system:

- When a new line extension is build, you will create a new service agreement that references your CIAC SA type. This service agreement should be linked to a service point associated with the extension. The easiest way to create this service

agreement and link it to the service point is by using the Start Account transaction. It's important to remember to define your CIAC SA type as a valid SA type for the SP type used to represent the line extension.

- When a new premise / service point is built that "hangs" off the original extension, the new service point should be linked to the original service point. This can be done by referencing the original service point as the "parent" service point of the new service point using a [foreign key value characteristic](#).
- The system periodically monitors the original CIAC service agreement. The process that performs this monitoring is referenced on the original CIAC service agreement's SA type. The exact processing that takes place during this monitoring is up to your organization's specific business requirement. For example, if you refund 10% of every bill produced for the "downstream" service points to the original contractor, you would have logic in your CIAC monitoring process that looks for recent bill segments produced for the "downstream" service agreements and then creates an adjustment for the master service agreement.

NOTE:

CIAC refunds are idiosyncratic. Because CIAC refund processing is idiosyncratic, we do not supply any CIAC monitoring processes in the base package. This is because the likelihood that they could be used is extremely low because of your organization's unique requirements.

Therefore, if your organization performs CIAC processing, you should create a special SA type.

CIS Division/ SA Type	Service Type	Distrib. Code	Debt Class	CIAC Process	Bill Seg Type
CA/CIAC	Other	A/R - CIAC	UNRE	Refund based on percent of future bills	Billable Charge

Loan Segmentation

If you loan money to customers that is recouped using an amortization schedule, you need to set up an SA type for the loan service agreement.

CIS Division/ SA Type	Service Type	Distrib. Code	Loan A/R Distrib. Code	Debt Class	Bill Seg Type
CA/LOAN	Other	A/R - LOAN	A/R - STLN	UNRE	LOAN

FASTPATH:

Refer to [Setting Up The System To Enable Loans](#) for more information.

Non-billed Budget Segmentation

If you allow your customers to pay set amounts at specified intervals (e.g. every two weeks), you need to set up SA types for non-billed budget service agreements. Non-billed budgets are typically used when your company bills on an infrequent basis and you want to provide your customers with a mechanism to make smaller payments more frequently. You may implement two types of non-billed budgets monitored and unmonitored, each type requiring a different SA type. You may also implement different renewal options for non-billed budgets.

CIS Division/ SA Type	Service Type	Distrib. Code	Non Billed Budget Monitoring	Debt Class	Renewal	Bill Seg Type
CA/NBB-MON	Other	A/P - OVPY	Monitored	NBB	Optional	Not billed
CA/NBB- UNMON	Other	A/P - OVPY	Unmonitored	N/A	Optional	Not billed

FASTPATH:

For more information about monitored and unmonitored non-billed budgets, refer to [Defining Non-billed Budget Options](#).

Designing SA Type For Other Segmentations

The earlier parts of this discussion described the most common factors that cause the creation of SA Types. However, many obscure factors could cause the introduction of more SA Types. In this section, we explain these more obscure factors.

WARNING:

We strongly recommend not being too pedantic when considering the factors described in this section. If you can only think of a few strange situations that would necessitate a SA type, think carefully before you introduce it. It's better to be a little less than perfect than end up with large number of obscure SA types.

Cash Distribution Segmentation

Every SA Type has a payment segment type. The payment segment type defines the cash account to which the SA type's payments should be booked. If different service agreements have different cash accounts, you will need to split the SA types accordingly.

Adjustment Profile Segmentation

Every SA Type has one or more adjustment profiles. These profiles define the valid adjustment types that can be booked to the SA type's service agreements. If different service agreements within an SA type have different mixtures of valid adjustment types, you must split the SA types accordingly.

Late Payment Charge Segmentation

An option exists on SA Type that causes the system to generate a late payment charge if payment is not received on time. If you don't levy late payment charges on all service agreements, you will need to determine when you do and design your SA types accordingly.

In addition, if you levy late payment charges, the percentage levied and the algorithm that defines the amount of the outstanding balance subject to the charge is defined on the SA type.

Debt Classification Segmentation

Every SA Type has a debt class. The debt class is used to categorize a service agreement's debt for the purpose of credit and collections (C&C) analysis. If a given SA Type has different categories of debt from C&C's perspective, you must split the SA Type.

FASTPATH:

For more information about debt class, refer to [Designing Your Collection Procedures](#).

NOTE:

Write Off Debt Class vs. Normal Debt Class. An SA type references both write off debt class and normal debt class. An SA type's write-off debt class controls the write-off rules imposed on service agreements of a given type. An SA type's normal debt class controls the collection rules imposed on service agreements of a given type. Refer to [Different Collection Criteria For Different Customers And Different Debt](#) for more information about collection rules. Refer to [Different Write-Off Criteria For Different Customers And Different Debt](#) for more information about write-off rules.

Allow Estimates Segmentation

Every SA Type has a switch that controls whether the system estimates consumption if meter reads are missing at billing time. If a given SA Type has different situations when the system should and should not estimate, you will have to split the SA Type.

NOTE:

Override. You can override the value of the SA Type's estimation switch on an individual service agreement. This means that if only a few service agreements don't abide by the SA Type's estimation switch, you can change the switch value of these service agreements.

FASTPATH:

For more information about estimation, refer to [Setting Up Consumption Estimation Parameters](#).

Severance Criteria Segmentation

Every SA Type has severance criteria. The severance criteria define the severance process used to sever service if the customer doesn't pay. You can have multiple severance processes if different conditions warrant a different process. For example, you may have a different severance process if the customer has life support.

If you have a SA Type that requires different severance conditions other than those currently supported, you can make a programmatic change to introduce the additional conditions OR you can split the SA Type.

FASTPATH:

For more information about severance, refer to [Designing Your Severance Procedures](#).

Deposit Class Segmentation

Every SA Type that exists to hold a cash deposit will reference a deposit class. The deposit class defines the business rules that control various functions including interest calculation and refund criteria. You will need multiple deposit SA Types if any of the deposit class' rules / conditions differ for different types of deposits. For example, if residential customers use a different recommended deposit algorithm as compared to commercial customers, you'd need one SA type for residential deposits and another for commercial deposits (where the residential deposit SA type will reference the residential deposit class and the commercial deposit SA type will reference the commercial deposit class).

You will need additional deposit SA types if your customers can have multiple deposits where each deposit is restricted to a specific type of debt. For example, if separate deposits are held for regulated and unregulated debt (and a customer could hold a combination of regulated and unregulated debt), you'd need one SA type for regulated deposits and another for unregulated deposits.

FASTPATH:

For more information about deposit class, refer to [Designing and Defining Deposit Classes](#).

Sub SA Types

If you operate in a deregulated environment AND if you provide billing services for other service providers, you will need to create SA types to handle the billing of the service providers' charges.

FASTPATH:

Refer to [Designing Your SA Types And Start Options For Sub SAs](#) for more information.

Financial Settlement SA Types

If you operate in a deregulated environment, you may have to create financial settlement service agreements for the service providers. As explained in [Service Providers Have Service Agreements Too, We Bill For Them and They Bill For Us](#) service providers require a service agreement to hold adjustments used to increase how much you owe the service provider (or how much they owe you).

FASTPATH:

Refer to [Designing SA Types For Service Provider Financial Settlements](#) for more information.

Interval Billing SA Types

If you have customers with interval meters, the SA types for these customers will require special setup.

FASTPATH:

Refer to [Designing Your SA Interval Billing Options](#) for more information.

Usage Request SA Types

If your organization uses a meter data management system to store meter reading information and the MDM is responsible for calculating bill determinants during billing, the SA types for these customers will require special setup.

FASTPATH:

For more information about usage requests, refer to [The Big Picture Of Usage Requests](#).

Initial Consumption Period Considerations

Bill segments produced for a service agreement have two time periods:

- The bill segment period. The bill segment period defines the entire period of time covered by a bill segment's charges.
- The consumption period. The consumption period defines the period of time used to calculate the number of days for daily charges.

The consumption period almost always starts one day after the bill segment period. The consumption period always ends on the bill segment's end date. For example, a bill segment period that spans 5-Jan-2002 through 6-Feb-2002 will almost always have a corresponding consumption period of 6-Jan-2002 through 6-Feb-2002. The reason that the start dates don't match is because a bill segment's start date equals the end date of the prior bill segment (i.e., the start date was already counted in the previous bill segment's consumption period and we don't want to count it twice).

The only time when the previous paragraph isn't true is the first bill segment that's produced for a new service agreement. This is because different utilities count the first day of a new service agreement differently than others. Because of this, a flag exists on SA Type called **Initial Start Date Option**. This flag controls whether the service agreement's start date is included in the consumption period in a service agreement's first bill segment.

The following table describes the ramifications of the options you can set for this flag.

Flag Value	Consumption Period Calculation	Use This Option When
Add 1 Day Always	The consumption period's start date is calculated by adding 1 day to the service agreement's start date. (The SA start date is never included in the consumption period for the first bill segment.)	You want the initial bill and all subsequent bills to have a consistently calculated consumption period (i.e., the consumption period for the first and all other bills is always one day less than the bill segment's period).
Add 1 Day for Back-to-back	A back-to-back occurs when any service point for this service agreement was previously linked to a different service agreement that was stopped on the same date that the new service was started (i.e., there is no gap in the service). If a back-to-back is detected, the consumption period start date is calculated by adding 1 day to the SA start date. If no back-to-back is detected, the start date of the consumption period is the SA start date.	You want to flexibly handle consumption period calculation. If you start customers on the same date as the stop date of the previous customer, billing does not include the SA start date in the consumption period. However, if you start a new customer one day (or more) after the stop date of the previous customer, billing includes the first day of the service agreement in the consumption period.
Include First Day	The start date of the consumption period is the service agreement start date. (The	You want to always include the first day AND you will never encounter a back-to-back situation.

SA start date is always included in the consumption period.)

The example below shows how the consumption period would be calculated with the various options for a customer who starts service on January 1.

	First Bill	Second Bill	Third Bill
	Meter Read: Jan 31	Meter Read: Feb 28	Meter Read: Mar 31
	Bill Segment Period: Jan 1 to Jan 31	Bill Segment Period: Jan 31 to Feb 28	Bill Segment Period: Feb 28 to Mar 31
Consumption period using Add 1 Day Always	Jan 2 to Jan 31 (30 days)	Feb 1 to Feb 28 (28 days)	Mar 1 to March 31 (31 days)
Consumption period using Add 1 Day for Back-to-back when back-to-back is detected	Jan 2 to Jan 31 (30 days)	Feb 1 to Feb 28 (28 days)	Mar 1 to March 31 (31 days)
Consumption period using Add 1 Day for Back-to-back when back-to-back is NOT detected	Jan 1 to Jan 31 (31 days)	Feb 1 to Feb 28 (28 days)	Mar 1 to March 31 (31 days)
Consumption period using Include First Day	Jan 1 to Jan 31 (31 days)	Feb 1 to Feb 28 (28 days)	Mar 1 to March 31 (31 days)

There may be SA types for which the value of this flag does not affect the consumption period calculation and still other SA types where this flag is never used. For example,

- For billable charge service agreements, the consumption period is equal to the start and end dates on the billable charge and therefore this flag is not applicable.
- A sub SA either inherits the consumption period from the master SA or it uses billable charges. As a result, billing does not use this flag.
- For some service agreements, the charges on the rate are not affected by the consumption period. For example, if you have a customer with a simple meter and a simple usage-based charge, billing calculates the amount of consumption between the start reading and end reading and applies the rate (i.e., the number of days in the consumption period doesn't impact the charges in some rates).
- Some service agreements are not billed, for example, [overpayment service agreements](#). For this type of service agreement and other types of service agreements that are not billed, this flag is not applicable. However, the system does not prevent a value from being entered in these cases to allow for an implementation process to use the flag if needed.

Setting this flag to an appropriate value is significant for certain types of service agreements.

- For services whose rate includes daily charges, the configuration of this flag may impact the first bill segment for the service agreement. For example, if the first bill period is October 1st through October 31st, do you consider consumption period to be 31 days or 30 days? How you want to bill the customer on the first bill determines how you set this flag.
- For interval billing service agreements, the interval usage recordings typically begin on the first day of service. If you add 1 to the start date, you miss billing for intervals on the first day. In this case, you should set the value to Add 1 Day for Back-to-back.

FASTPATH:

Refer to [Determine the Consumption Period](#) for more information.

Processing Sequence Considerations

You may have customers with a complex collection of contracts such that the calculation for one bill segment relies on information calculated by another bill segment for the same account. For example, perhaps you need to process your bill segments as follows:

- Step 1: Calculate bill segments for all the account's "single site" service agreements (i.e., each service agreement related to a single premise)
- Step 2: Calculate the bill segment for additional charges for the "head office" service agreement where the charges are based on the aggregated consumption of all premises from all bill segments (calculated in the first step).

To create the "head office" bill segment for the account, you must control the order in which the system creates the bill segments for each service agreement.

The SA type allows you to indicate a billing processing sequence that controls the order in which the service agreements are processed. The processing sequence is optional and service agreements are processed in ascending order of their SA type's billing processing sequence. As a result, SA Types with a processing sequence of zero are processed first, then 1, then 2, etc.

NOTE:

If you have sub SAs linked to a master SA, the billing processing sequence is used first to order the creation of bill segments for the master SAs. If you populate a processing sequence on an SA type for a sub SA, it is used to control the order in which the sub SAs for a given master SA are processed relative to each other. Refer to [Designing Your SA Types And Start Options For Sub SAs](#) for more information.

The billing processing sequence also controls the order of service agreements in the following other processes:

- Execution of pre bill completion algorithms. The system processes each service agreement in the billing processing sequence order. Within each service agreement, the pre bill completion algorithms on its [SA type](#) are processed in the order of the algorithm's sequence.
- Execution of bill completion algorithms. The system processes each service agreement in the billing processing sequence order. Within each service agreement, the completion algorithms on its [SA type](#) are processed in the order of the algorithm's sequence.
- Interval Data Creation. For interval service agreements linked to interval profiles with profile creation algorithms defined, the system processes each service agreement in the processing sequence order. Within each service agreement, the [data creation algorithms](#) are processed in the order of the creation priority on the [profile type](#).

Designing Prepayment Billing Options

This section provides guidelines describing how to design service agreements that handle smart meter prepaid billing.

When a customer is subject to prepayment, normal cycle-based billing is not used to calculate the bill segments for their service. Rather, a prepay billing (PPB) process creates bill segments frequently (for example, daily) and periodically requests funds to maintain an adequate payoff balance. Bills will only be used to periodically inform the customer of the various financial transactions that have occurred since the last bill was produced.

Implementing prepayment billing requires the configuration of the following objects:

- SA Type
- Adjustment Types
- Bill Segment Type

- Service Task Type

These configuration tasks are described below.

NOTE: For additional information, see the detailed descriptions in the application for the C1-PrepayBillerTaskType and C1-PrepayBillerTask business objects.

SA Type for Prepay Billing

To configure a prepay billing SA Type, create a new SA type with the following attributes:

- Set the Special Role flag to **Bill Determinants Required**. The Prepay Option should be set to **Prepaid Smart Meter**.
- Specify the SA Creation system event with the algorithm CI_CREATEPPB to create the service agreement's prepay biller.
- Specify the Payment Freeze system event with the algorithm STPZ-RMVC to set current amount to zero on a payment.
- If required, specify the Bill Completion system event with an algorithm of the type C1-CREPPBBM to add a bill message that informs an end customer of their prepaid charges for the period. You may need to create an algorithm of this type if it does not exist in your system.
- **Eligible for Budget** should be set to false.
- Reference a new **Prepay Bill Segment Type**, which is described below.

The following adjustment types must also be defined on the prepay SA type:

- **Adjustment Type (Xfer)** specifies the type of adjustment used to transfer funds from the delinquent service agreements to the prepaid service agreement
- **Adjustment Type (Current=0)** specifies the type of adjustment used to set the prepaid current balance to zero after debt has been transferred
- **NSF Adjustment Type** specifies the type of adjustment used to levy an NSF charge. The NSF Adjustment Type is described in greater detail below.

The new SA types must be added to the data value mapping between CCB and MDM. This is to enable the synchronization of SAs of this type to MDM. The new CCB SA types can be mapped to existing US types in MDM. For additional information about this configuration, please see the integration documentation.

Adjustment Types for Prepay Billing

- **Funds Request Adjustment Type:** This adjustment type captures the type of adjustment used to request more prepaid funds. This adjustment type is specified on the prepaid biller task type. Specify the following algorithms on this adjustment type:
 - **Adj. Financial Algorithm** should reference the Adjustment FT creation algorithm ADJT-CA (Payoff=0/Current=Adj Amount (No GL)).
 - **Adjustment Freeze Algorithm** should reference an algorithm of the type C1-PPBADJFRZ (Set Prepaid Funds Request FT Details). You may need to create an algorithm of this type if it does not exist in your system.
- **NSF Adjustment Type:** For prepaid customers on autopay, when a user cancels an autopay tender the cancel reason must indicate that an NSF charge should be levied. The system invokes the NSF Charge algorithm specified on the tender's account's customer class which creates the NSF adjustment on the appropriate SA. The charge is levied using the NSF Adjustment Type specified on the SA's SA type. This adjustment type must reference the new adjustment freeze algorithm that will disable automatic payments on the PPB. This algorithm must be based on the algorithm type C1-DISPPBAP (Disable PPB's Automatic Payment). You may need to create an algorithm of this type if it does not exist in your system.

Bill Segment Type Required for Prepay Billing

Configure a new bill segment type for prepaid service agreements. This bill segment type should have the following attributes:

- **Create Algorithm** should be set to C1-BSBS-UR to create bill segment using a usage request.
- **Financial Algorithm** should be set to a new algorithm with algorithm type C1-NEM-GL (Payoff Amt = Bill Amt / Current Amt = 0 (GL affected)). You may need to create an algorithm of this type if it does not exist in your system.
- **Get Consumption Algorithm** should be set to C1-BSGC-USG to get bill segment consumption using a usage request.

Service Task Type for Prepay Billing

Configure a new service task type for prepay billing. This task type should reference the Prepay Biller (PPB) Task Type Business Object (C1-PrepayBillerTaskType). The task type must be specified on the new prepay billing SA creation algorithm. The task type captures the adjustment type used when an adjustment is created to request additional funds from the customer. Ensure that this adjustment type references the adjustment freeze algorithm (C1-PPBADJFRZ) that populates the financial transaction's arrears date.

SA Types And The Financial Design

In this section, we provide an example of how our SA Types map to Bill Segment Types, Payment Segment Types, and Adjustment Profiles. This example is meant to help solidify the power of the financial model, it is not necessarily indicative of how your specific implementation will look.

WARNING:

If you are not comfortable with the topics described in [Defining Financial Transaction Options](#), the following table will not make sense.

Division/SA Type	Distribution Code	Bill Segment Type	Payment Segment Type	Adjustment Profiles
CA/G-RES	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES, BUDGET
CA/G-COM	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/G-IND	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/W-RES	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/W-COM	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/W-IND	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/E-RES	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES, BUDGET
CA/E-COM	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/E-IND	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/WW-RES	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/WW-COM	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/WW-IND	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/CABLE	A/R-UTIL	SP-RATED	NORMAL	BALXFER, MISCFEES
CA/E-COY	E-COMP	COMPUSAG		
CA/G-COY	E-COMP	COMPUSAG		
CA/W-COY	E-COMP	COMPUSAG		
CA/VO-STD			NORMAL	BALXFER
CA/VO-LIA			NORMAL	BALXFER
CA/CONNECT			NORMAL	BALXFER, MISCFEES

CA/CHARITY	A/P-CHAR	RECUR	CHARITY	CHARITY
CA/PA-REGU	A/R-ARRG	RECUR-AS	NORMAL	BALXFER, MISCFEES, DPA
CA/PA-UNRE	A/R-ARRG	RECUR-AS	NORMAL	BALXFER, MISCFEES, DPA
CA/MERCH-I	A/R-MRCH	RECUR-AS	NORMAL	BALXFER, MISCFEES, MERCH
CA/DEP-I	A/P-DEPO	RECUR-AS	NORMAL	BALXFER, MISCFEES, DEPOSIT
CA/ONETIME	A/R-UTIL	BILLCHRG	NORMAL	BALXFER, MISCFEES
CA/OVR UNDR	EXP-OV/UND		NORMAL	
CA/OVERPAY	A/P-OVER		NORMAL	BALXFER
CA/SUSPENSE	A/R-SUSP		NORMAL	
CA/NBB	A/P-OVPY		NORMAL	BALXFER, MISCFEES, NBB

If you operate in a deregulated environment, you will also have additional SA types as described under [Designing Your SA Types And Start Options For Sub SAs](#) and [Designing SA Types For Service Provider Financial Settlements](#) you may have additional SA types.

Setting Up SA Types

In the previous section, [Designing SA Types](#), we presented a case study that illustrated a mythical organization's SA types. In this section, we explain how to use the windows on the SA Type window group to maintain your SA Types.

NOTE:

When a new SA type is added. When you add a SA type whose service agreements use service points, you must update the respective SP types if the new SA type is defaulted when service is initially started at the service points of a given SP type. Refer to [SP Type - SA Type](#) for more information.

SA Type - Main Information

Open **Admin > Customer > SA Type > Add** to define core information about your SA Types.

WARNING:

Every SA Type is owned by a CIS Division. This Division controls many values that can be referenced on the SA Type. If you don't understand Divisions and their place in the application, do NOT attempt to set up your SA Types. Rather, refer to [Setting Up CIS Divisions](#) before proceeding.

Description of Page

Enter a unique combination of **CIS Division** and **SA Type** for every service agreement type.

Enter a **Description** for the SA type.

Service Type defines the type of service associated with the SA type. If the SA type has rates, only rates belonging to this service type may be linked to the SA type.

FASTPATH:

For more information about service types, refer to [Setting Up Service Types](#).

Select the **Distribution Code** and **GL Division** that defines the receivable account for receivable-oriented service agreements. For non-receivable oriented service agreements, this distribution code is typically as follows:

- Charitable contributions. The distribution code is a charity payable account.
 - Deposits. The distribution code is a deposit payable account.
 - Non-billed budgets. The distribution code is an overpayment payable account.
 - Company usage. The distribution code is a company usage expense account.
 - Write off. The distribution code is a write-off expense account.
 - Payment arrangements. The distribution code is a payment arrangement receivable account.
-

FASTPATH:

For more information about GL accounts, refer to [The Source Of GL Accounts On Financial Transactions](#).

Select the **Revenue Class** associated with the SA Type (and its service agreements). The revenue class may affect the revenue account(s) generated by the service agreement's rate.

FASTPATH:

Refer to [Designing Calculation Groups and Rules](#) for more information about revenue class.

Turn on **Start Options Required** if you want to force a customer service rep to choose a start option when they start service for this SA Type (on the Start Account window). If this switch is off and a rate is required for the SA Type, the system defaults the SA type's default rate on new service agreements. The default rate is defined on the SA Type - Rate page.

FASTPATH:

Refer to [Setting Up Start Options](#) for more information about the pros and cons of requiring start options.

Select the **Pay Segment Type** that defines how payment segments linked to service agreements of this type affect:

- The service agreement's payoff and current balances
-

FASTPATH:

For more information about payment segment types, refer to [Setting Up Payment Segment Types](#).

When a tender is canceled, a cancellation reason must be supplied. If the cancellation reason indicates a NSF (non sufficient funds) charge should be levied, the system invokes the Levy an NSF Charge algorithm specified on the tender's account's [customer class](#). Because adjustments must be linked to a service agreement, the algorithm must determine the appropriate service agreement to use to levy the adjustment based on business rules. The charge is levied using the **NSF Adjustment Type** of the appropriate service agreement's SA type.

WARNING:

You must specify adjustment type profiles on the SA type (on the Adjustment Type window) before adjustment types will appear in the above drop downs.

FASTPATH:

For more information about adjustment types, refer to [Setting Up Adjustment Types](#). For more information about cancellation reasons, refer to [Setting Up Payment Cancellation Reasons](#).

Select the **Payment Priority**. This field is available for use by the algorithms that distribute partial payments amongst an account's service agreements. Higher priority service agreements will have their debt relieved before lower priorities. Refer to [Distribution Based on Payment Priority](#) and [Delinquent Payment Distribution Algorithm](#) for information about payment distribution algorithms that use this field.

NOTE:

The values for this field are customizable using the Lookup table. This field name is PAY_PRIORITY_FLG.

FASTPATH:

For more information about distribution priority, refer to [Distributing A Payment Amongst An Account's Service Agreements](#).

Select the **Delinquent Payment Priority**. This field is available for use by the algorithms that distribute partial payments amongst an account's service agreements. Higher priority service agreements will have their debt relieved before lower priorities. Refer to [Delinquent Payment Distribution Algorithm](#) for information about a payment distribution algorithm that uses this field.

NOTE:

The values for this field are customizable using the Lookup table. This field name is DEL_PRIORITY_FLG.

Turn on **Do Not Overpay** if the system is not allowed to distribute an overpayment to this type of service agreement (i.e., the service agreement is not allowed to have a system-created credit balance). This field is available for use by algorithms that distribute overpayments. Refer to [Overpayments Held On Highest Priority Service Agreement](#) for information about an overpayment algorithm that uses this field.

Turn on **Late Payment Charge** if the system should generate a late payment charge for this type of service agreement if payment is not received on time. If this is turned on, you must define the **LPC Calc. Algorithm** used to calculate the late payment charge amount. Refer to [Defining Late Payment Charge Options](#) for more information about late payment charges. Examples of algorithm types used for calculating late payment charges are [BILPC-SPRC](#) and [BILPC-TOTAL](#).

Define the **Adjustment Type (Synch Curr)** that will be used to synchronize (make equal) the current amount with the payoff amount on a service agreement of this type. This type of processing happens as follows:

- Most [write-off](#) algorithms that perform financial efforts (e.g., writing off debt), will issue an adjustment of this type if the service agreement's current and payoff balances are not equal.
- If a user stops a customer on a [budget plan](#), the system issues adjustments of this type to synchronize the customer's current and payoff balances.
- If a user stops a service agreement covered by a [non-billed budget](#), the system issues adjustments of this type to synchronize the customer's current and payoff balances.
- If a [cancellation of a bill segment](#) occurs after a customer has stopped participating in a budget plan, an adjustment of this type is issued to synchronize the imbalance created when the bill segment's financial transaction is canceled.

Turn on **CIAC SA Type** and specify an appropriate **CIAC Refund Process** if service agreements of this type are used to bill for Contribution In Aid of Construction (CIAC) charges. Refer to [CIAC Segmentation](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SA_TYPE](#).

SA Type - Detail

Open **Admin > Customer > SA Type > Search** and navigate to the **Detail** page to define additional details about a given SA type.

Description of Page

Turn on **Display As Alert** if Control Central should display an alert if an account has a service agreement of this type that isn't Closed or Canceled. If this switch is on, also enter the **Alert Information** to appear on Control Central. We recommend only using this feature on unusual SA types (e.g., payment arrangements, write-offs) so that a CSR is not presented with an alert for every SA type.

If this SA Type is used for any of the **Special Roles**, defined in the dropdown, indicate which one. Valid values are: Billable Charge, Cash Deposit, Interval, Loan, Non-billed Budget, Payment Arrangement, Write Off, Bill Determinants Required. This information is used on windows with functionality that can only be used by service agreements used for specific roles. For example, the Billable Charge window group can only reference Billable Charge service agreements.

If Special Role is Cash Deposit, you must define the **Deposit Class** of the deposit. You should also define a **Deposit Class** on every SA type to which a given deposit can be distributed.

FASTPATH:

Refer to [What Do Deposit Classes Do?](#) for more information.

If the Special Role is Loan, you must also define the following fields:

- Use the **Interest Bill Factor** to define the bill factor code for the loan interest rate.
- Use **Override Interest Flag** to indicate whether the interest rate defined on the interest bill factor may be overridden at the SA level. If you select Allowed, the interest rate may be overridden by a contract value on a start option or the SA.
- Use the **Loan A/R Distribution Code** to define the distribution code to be used when posting the short-term receivable amount to the general ledger (the normal distribution code is used for the long-term receivable). If the normal distribution code is the same as the **Loan A/R Distribution Code**, the SA type does not differentiate between long- and short-term receivables. If the two distribution codes are different, the SA type differentiates between long- and short-term receivables.

FASTPATH:

Refer to [Defining Loan Options](#) for more information about **Interest Bill Factor**, **Override Interest Flag** and **Loan A/R Distribution Code**.

If the Special Role is Non-billed Budget , you must also define the following:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to the transfer accumulated credit from the non-billed budget SA to the SAs covered by the non-billed budget when the account is billed or the non-billed budget SA is stopped.
- Use the **Non-billed Budget Monitoring** to specify whether the non-billed budget is monitored by the account debt monitor.

If the SA type is defined as Eligible for Non-billed Budget, you must also define the following:

- Use **Adjustment Type (Current=0)** to specify the type of adjustment used to set the service agreement's current balance to zero when a service agreement of this type is linked to an active, monitored non-billed budget.

If the Special Role is Payment Arrangement, you must also define two adjustment types:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to transfer funds from the delinquent service agreements to the payment arrangement service agreement.

- Use **Adjustment Type (Current=0)** to specify the type of adjustment used to set the payment arrangement's current balance to zero after funds have been transferred.

If the Special Role is Write Off, you must also define the following adjustment types:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to transfer funds from the uncollectable service agreements to the write off service agreement.

If the Prepay Option is **Prepaid Smart Meter**, you must also define two adjustment types:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to transfer funds from the delinquent service agreements to the prepaid service agreement.
- Use **Adjustment Type (Current=0)** to specify the type of adjustment used to set the prepaid current balance to zero after debt has been transferred.

WARNING:

You must specify adjustment type profiles on the SA type (on the Adjustment Type window) before adjustment types will appear in the above drop downs.

The **Stop Option Flag** can be used to automatically stop a service agreement when all other service agreements of an account are stopped. Valid values are: Automatically Stop SA. When all service point related service agreements of an account are stopped then all additional service agreements where the **Stop Option** is set to Automatically Stop SA will also be stopped. This is useful, for example, with charitable contribution service agreements. The charity service agreement should be stopped when all utility related services are stopped.

FASTPATH:

Refer to [The Lifecycle of a Service Agreement](#) for more information on the **Stop Option**.

Turn on **One Time Charge** if this SA type is used for one-time invoices. When a one-time invoice service agreement is created, the system sets the stop date of the SA to be equal to the start date.

Turn on **Sub SA** if this SA type is used to define the business rules for sub service agreements.

FASTPATH:

Refer to [Sub Service Agreements](#) for more information about sub service agreements.

Renewal of SAs of this type may be Optional, Not Allowed or Required depending on your business processes. If renewal is not allowed, the SA expires on the expiration date. Renewal treatment is an important consideration for SAs that require an expiration date, such as [non-billed budget SAs](#).

If renewal is required or optional, specify the **Days Before Expiration for Renewal**. Note that currently this is only used by non-billed budgets to calculate the renewal date based on the expiration date.

If the Special Role is Non-billed Budget, **Non-billed Budget Monitoring** must indicate whether the non-billed budget is monitored by the account debt monitor.

FASTPATH:

Refer to [Credit and Collections and Non-billed Budgets](#) for more information about monitoring non-billed budgets.

Where Used

The alert information is used by Control Central to alert a CSR when unusual service agreements exist for an account. Refer to [Control Central - Main](#) for more information.

Only SA types designated as being Billable Charge may have billable charges linked to them. Refer to [Maintaining Billable Charges](#) for more information.

Only SA types designated as being Cash Deposit are processed by the various deposit-related background processes (e.g., interest calculation, automatic refund, etc.). Refer to [The Big Picture Of Deposits](#) for more information.

Only SA types designated as being Interval may define Contract Option Types, Profile Relationship Types and TOU Map Types. Refer to [Designing Your SA Interval Billing Options](#) for more information. This role also ensures that a service agreement of this type defines the cutoff time and start day option required by billing. Refer to [Start and End Times for Billing](#) for more information.

Only SA types designated as Loan are used to define the loan terms for a loan SA. Refer to [Loans](#) for more information.

Only SA types designated as Non-billed Budget may be used to set up non-billed budgets. Refer to [Non-billed Budgets](#) for more information.

Only SA types designated as being Payment Arrangement may be used on the payment arrangement window group. Refer to [Setting Up Payment Arrangements](#) for more information.

Only SA types designated as being Write Off may be specified as the write off SA type on distribution codes. Refer to [Setting Up Distribution Codes](#) for more information.

Only service agreements whose SA type is designated as being Write Off appear on the Write Off SAs query. Refer to [Write Off - Write Off SAs](#) for more information.

SA Type - Billing

Open **Admin > Customer > SA Type > Search** and navigate to the **Billing** page to define how the system manages bill segments for service agreements of a given SA type.

Description of Page

Turn on **Eligible for Billing** if the system should create bill segments for service agreements of this type. This will typically be turned on for all service agreements except for those used to hold write-off amounts or to levy one-off adjustments.

Define the minimum number of days a bill segment (other than the final segment) must span using **Minimum Days for Billing**. This is useful to prevent initial bill segments that span only a few days.

FASTPATH:

For more information about minimum days, refer to [Preventing Short Bill Segments](#).

Select the **Bill Segment Type** that controls both how bill segments for this SA Type will be created and how the related financial transaction affects the general ledger and the customer's debt.

FASTPATH:

For more information about bill segment types, refer to [Setting Up Bill Segment Types](#).

Use **DefaultDescription on Bill** to define the verbiage that should print on the customer's bill.

NOTE:

Rates overwrite this description. The Default Description on Bill is not applicable for service agreements whose charges are calculated using a rate. Why? Because the description that appears on the bill segment is defined on the rate schedule's rate version calculation group.

Billable charges overwrite this description. The Default Description on Bill is not applicable for service agreements whose charges are calculated using a billable charge. Why? Because the description that appears on the bill segment is defined on the billable charge.

Use the **Billing Processing Sequence** if you need to control the order in which service agreements linked to this SA type are processed by billing and interval data creation processes.

FASTPATH:

Refer to [Processing Sequence Considerations](#) for more information.

Use **Bill Print Priority** to define the order in which the SA type's bill segments should appear on bills (relative to the other SA types that appear on a bill).

NOTE:

The values for this field are customizable using the Lookup table. This field name is BILL_PRT_PRIO_FLG.

Use **Max Bill Threshold** if you want the system to generate a bill error when a bill segment is produced in batch that exceeds a given value. These bill errors will appear on the standard billing queries and To Do lists. If, after reviewing the high value bill segment, an operator truly intends to send the bill out, they should regenerate the bill. Refer to [How To Correct A Bill Segment That's In Error](#) for more information.

WARNING:

The value entered in this field will DEFAULT onto service agreements of this type when they are first created. An operator may change the default value on a service agreement in case a specific customer has unusually high bills that continually error out. It's important to be aware that if you change the value of High Bill Amount on an SA type and there already exist service agreements of this type, the existing service agreements will contain the original value (the new value on the SA type will not be propagated on the existing service agreements).

Use **Graph Unit Of Measure** to define the unit of measure of the graphed consumption on the bill (if any).

Turn on **Allow Estimates** if the system is allowed to generate estimated consumption if meter reading(s) cannot be found at billing time. This value is defaulted onto service agreements and can be overridden on an individual service agreement.

FASTPATH:

For more information about estimated consumption, refer to [The Theory Behind Consumption Estimation](#).

Turn on **Characteristic Premise Required** if a characteristic premise must be linked to the service agreement when the service agreement is activated. The characteristic premise is used to define the taxing authorities associated with the service agreement's bill segments. It is also used to identify where the service agreement's service is located on various windows.

FASTPATH:

For more information about how characteristic premise is used, refer to [An Illustration Of A Bill Factor And Its Characteristics](#).

Use the **Initial Start Date Option** to control how billing should calculate the consumption period for the very first bill for service agreements of this type. This field is not applicable for sub SA types or SA types with a special role of Billable Charge. Valid values are Include First Day, Add 1 Day Always and Add 1 Day for Back-to-back. Refer to [Initial Consumption Period Considerations](#) for more information.

Non-metered service agreements may have the end date of their bill segments defined on a user-maintained bill period schedule. This option is used when bill segments must fall on strict calendar boundaries (e.g., quarterly bills that end on the last day of the quarter). If this SA type should be billed like this, select Use Bill Period in the **Use Calendar Billing** field. When this option is used, you must define the **Bill Period** whose schedule defines the bill segment end dates.

FASTPATH:

For more information about bill period schedules, refer to [Designing Bill Periods](#). For more information about other bill end date methods, refer to [Ways To Control The End Date Of A Bill Segment](#).

Instead of the Use Bill Period method, non-metered service agreements may have their bill segment end date based on the first day of service. For example, if service started on the 16th of some month, the ongoing bill segments will start on roughly the 16th of each month. This option is frequently used to bill for garbage or cable service. If this SA type should be billed like this, select Anniversary Future Billing or Anniversary Past Billing in the **Use Calendar Billing** field. When either option is used, you must define the **Anniversary Bill Frequency**. This frequency defines the amount of time between bill segments.

FASTPATH:

For more information about anniversary billing, refer to [Using The Anniversary Method](#). For more information about other bill end date methods, refer to [Ways To Control The End Date Of A Bill Segment](#).

Total Bill Amount indicates whether service agreements of this type can use the total amount to bill field on the service agreement page. Valid values are Not Allowed and Required. Only SA types used to bill for deposits or loans should have this field set to Required.

If Required is selected, you must enter the **Total Amount To Bill Label**. The **Total Amount To Bill Label** defines the label that prefixes the total bill amount on the service agreement page for service agreements of this SA type.

FASTPATH:

For more information about total amount to bill and deposit service agreements, refer to [Total Amount To Bill](#). For more information about total amount to bill and loan service agreements, refer to [Setting Up The System To Enable Loans](#).

Recurring Charge indicates whether service agreements of this type can use the recurring charge field on the service agreement window. Valid values are Not Allowed, Optional and Required . If either Optional or Required are used, you must enter:

- **Recurring Chg Amt Label**. This defines the label that prefixes the recurring charge amount on the service agreement window for service agreements of this SA type.
- **Recurring Charge Frequency**. This defines the following:
 - Specifies the frequency at which the Recurring Charge Amount specified on service agreements of the SA Type is to be billed.
 - Serves as the basis for proration of the Recurring Charge Amount.
 - Specifies the frequency at which service agreements of the SA Type without a rate and/or meters will be billed.

FASTPATH:

For more information about how to use the recurring charge information, refer to [Charitable Contribution Segmentation](#), [Merchandise Segmentation - Installment Billing](#), [Deposit Segmentation - Installment Billing](#), [Payment Arrangement Segmentation](#), [Budget Billing Segmentation](#), [The Terms Of A Loan Are Stored On A Service Agreement](#).

Turn on **Eligible for Budget** if service agreements of this type can participate in budget billing. If this switch is turned on, then you must define the **Adjustment Type (Synch Current)** that will be used to synchronize (make equal) the current amount with the payoff amount on a service agreement of this type when a budget is cancelled. (The Adjustment Type (Synch Current) field is on the main page.)

FASTPATH:

Refer to [Budget Billing](#) for more information about budgets in general. Refer to [Budget Billing Segmentation](#) and [Designing and Defining Budget Plans](#) for more information.

Set the **Eligible for Non-billed Budget** flag to Eligible for Non-billed Budget if you want SAs of this type to be eligible to be covered by a non-billed budget. If this flag is set to Eligible for Non-billed Budget, you must also define the **Adjustment Type (Current = 0)** field (on [SA Type - Detail](#)).

FASTPATH:

Refer to [Current Amount For SAs Covered By A Non-billed Budget](#) and [SA Types for SAs Covered by Non-billed Budgets](#) for more information.

Require Total Amount Switch versus Bill Segment Algorithm

The following table shows valid combinations of the SA type's required total amount switch and the bill segment creation algorithm defined on the SA type's bill segment type. If N/A appears in a cell, the combination is not supported in the system. Otherwise, we list typical types of service agreements that will use a combination.

Bill Segment Create Algorithm	SA Type Require Total Amount Switch:	SA Type Require Total Amount Switch:
	Not allowed	Required
Apply Rate	Metered services. Lamp services. Misc item services. Company usage. Misc recurring charges whose value is specified in a rate or is taxable.	N/A
Recurring Charge With Auto Stop	Payment arrangements. Merchandise installment plans. Zero-interest loans.	N/A
Recurring Charge For Amount To Bill	N/A	Deposit installment plans.
Recurring Charge	Charitable contributions.	N/A
Billable Charge	One time invoices. Pass through charges	N/A
Loan	N/A	Loans.

Allow Recurring Charge Switch versus Bill Segment Algorithm

The following table shows valid combinations of the SA type's allow recurring charge switch and the bill segment creation algorithm defined on the SA type's bill segment type. If N/A appears in a cell, the combination is not supported in the system. Otherwise, we list typical types of service agreements that will use a combination.

Bill Segment Create Algorithm	SA Type Recurring Charge Switch:	SA Type Recurring Charge Switch:	SA Type Recurring Charge Switch:
	Not allowed	Optional	Required
Apply Rate	Metered services - no budget Lamp services Misc item services. Company usage.	Metered services - budget optional	Metered services - budget required

Misc recurring charges whose value is specified in a rate or is taxable.

Recurring Charge With Auto Stop	N/A	Payment arrangements. Merchandise installment plans. Zero-interest loans. SEE NOTE!	Payment arrangements. Merchandise installment plans. Zero-interest loans. SEE NOTE!
Recurring Charge For Amount To Bill	N/A	Deposit installment plans. SEE NOTE!	Deposit installment plans. SEE NOTE!
Recurring Charge	Charitable contributions	N/A	N/A
Billable Charge	One time invoices. Pass through charges	N/A	N/A
Loan	N/A	N/A	Loans.

NOTE:

Most recurring charge SA types require a recurring charge amount on their service agreements. However, the above matrix indicates you can have recurring charge SA types where this value is optional. Why? A special algorithm exists in billing that says if the recurring charge amount is 0 (zero) the system will bill the remaining payoff balance or total amount to bill. This algorithm exists so that you can easily bill the amount in one lump sum (i.e., don't bill it in installments).

Where Used

The billing information is used when the system creates a bill segment for service agreements of this type.

SA Type - Rate

Open **Admin > Customer > SA Type > Search** and navigate to the **Rate** page to define the rates that may be referenced on service agreements of a given type.

Description of Page

Turn on **Rate Required** if the bill segment creation algorithm for the SA type expects a rate schedule to be referenced on service agreements of this type.

FASTPATH:

For more information, refer to [Rates](#).

Define the date the system uses when selecting an effective-dated rate (from the service agreement's rate history) using **Rate Selection Date**. Selecting Bill Start Date will cause the system to use the rate effective on the first day of the bill segment's [consumption period](#). Selecting Bill End Date will cause the system to use the rate effective on the last day of the bill period. Selecting Accounting Date will cause the system to use the rate effective on the accounting date associated with the bill.

If the Contract Management module is not [turned off](#), which includes [umbrella agreement management](#), indicate the **Rate Source**. Check SA Only indicates that the rate schedule currently in effect for the service agreement should always be used. Check TOS First, then SA indicates that if the service agreement is linked to a terms of service record, the service agreement's rate could be [overridden by one linked to the terms of service](#) record.

The information in the **Rate Schedules** collection defines the rates that may be referenced on service agreements of this type. The following fields are required for each SA Type:

Rate Schedule Specify the rate schedule; its description is displayed adjacent.

Use As Default Turn on this switch for the rate to be defaulted on new service agreements.

Rate Required versus Bill Segment Algorithm

The following table shows appropriate combinations of the SA type's rate required switch and the bill segment creation algorithm defined on the SA type's bill segment type. If N/A appears in a cell, the combination is not applicable. Otherwise, we list typical types of service agreements that will use a combination. Be aware that no cross validation exists between the rate required switch and the bill segment creation algorithm when setting up the SA type.

Bill Segment Create Algorithm	SA Type Rate Required Switch:	SA Type Rate Required Switch:
	Not allowed	Rate required on SA
Apply Rate	N/A	Metered services. Lamp services. Misc item services. Company usage.
Recurring Charge With Auto Stop	Payment arrangements. Merchandise installment plans. Zero-interest loans.	N/A
Recurring Charge For Amount To Bill	Deposit installment plans.	N/A
Recurring Charge	Charitable contributions.	N/A
Billable Charge	One time invoices.	Billable charges that require a rate to add-on extra charges (like taxes) or billable charges where the consumption is interfaced and the system is responsible for calculating the charges.
Loan	Loans.	N/A

Where Used

This information is used to default and validate the rate specified on a service agreement. Refer to [Service Agreement - Rate Info](#) for more information.

SA Type - SP Type

Open **Admin > Customer > SA Type > Search** and navigate to the **SP Type** page to define the service point types that may be referenced on service agreements of a given type.

Description of Page

Turn on **Service Points Required** if at least one service point should be linked to service agreements of this type in order to properly bill the service agreements.

The information in the **SP Types** collection defines the service point (SP) types that may be referenced on this SA type's service agreements. The following fields are required for each SA Type:

SP Type Specify the SP type; its description is displayed adjacent.

FASTPATH:

For more information about SP types, refer to [Designing SP Types](#).

Where Used

This information is used to validate the types of service points linked to a service agreement. Refer to [Service Agreement - SA / SP](#) for more information.

SA Type - Adjustment Profiles

Open **Admin > Customer > SA Type > Search** and navigate to the **Adj Profile** page to define the adjustment profiles that define adjustment types that may be referenced on service agreements of a given type.

Description of Page

Define the **Adjustment Type Profiles** that, in turn, define adjustment types that may be referenced on service agreements of a given type.

FASTPATH:

For more information about adjustment type profiles, refer to [Setting Up Adjustment Type Profiles](#).

Where Used

This information is used to validate the adjustments linked to the service agreement. Refer to [Adjustments - Main Information](#) for more information.

SA Type - C&C

Open **Admin > SA Type > Search** and navigate to the **C&C** page to maintain attributes that affect how the system severs the service agreement when collection attempts fail.

FASTPATH:

Refer to [Designing Your Severance Procedures](#) for more information.

Description of Page

Select the **Debt Class** associated with the SA Type. Any debt on a service agreement of this SA Type will be categorized under this debt class.

Select the **Write Off Debt ClassCode** associated with the SA Type. Any debt on a service agreement of this SA Type will be categorized under this debt class during write-off processing.

NOTE:

Write Off Debt Class vs. Regular Debt Class. It's important to be aware that a SA type references both a regular debt class and a write-off debt class. The regular debt class controls the collection criteria applied against an account's service agreements. The regular debt class is also used to segregate an account's outstanding balance on several queries in the system. The write-off debt class controls the write-off criteria applied against an account's stopped service agreements. The reason the system supports two different debt classes is because you may categorize your service agreements differently when you try to collect overdue debt versus when you write-off debt. Refer to [The Big Picture Of Write Off Processing](#) for more information.

The information in the **Severance Criteria** collection defines the SA Type's severance criteria. Severance criteria define the severance process to be executed for service agreements of a given SA type. The severance process may differ depending

on some attribute of the customer or premise. For example, you may have a different severance process if the customer has life support equipment.

FASTPATH:

The following information is not intuitively obvious. Refer to [Designing Your Severance Procedures](#) for more information.

The following fields are required for each instance:

Priority The priority controls the order in which the system determines if the severance process should be applied (the first severance process whose algorithm applies is used). Higher priorities are checked before lower priorities.

NOTE:

The values for this field are customizable using the Lookup table. This field name is CRIT_PRIO_FLG. Be aware that this field is used for multiple tables: [Collection Class Control](#), [Severance Criteria](#), [Write Off Control](#) and [Workflow Process Profiles](#).

Severance Criteria Algorithm Select the algorithm to be used to check if the severance process should be initiated for service agreements of this type. If a condition is met, a severance process is created using the associated severance process template.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
 - On this algorithm, reference an Algorithm Type that determines if the severance of a service agreement should be processed using the associated **Severance Process Template**. Click [here](#) to see the algorithm types available for this plug-in spot.
-

IMPORTANT:

You must have at least one entry in this collection otherwise the system will not start a severance process when a service agreement of this type needs to be stopped due to non payment. This entry should have the lowest priority code and should reference a **Severance Criteria Algorithm** that references the [SV CRIT DFLT](#) algorithm type.

Severance Process Template Specify the severance process template to use to sever the service agreement; its description is displayed adjacent.

Where Used

The debt class has multiple uses:

- The system summarizes an account's debt by debt class on [Account - Main Information](#) and [Account - Financial Balances](#).
- Debt class is one component that controls how the system analyzes an account's overdue debt (the others are the account's collection class and currency). Refer to [Different Collection Criteria For Different Customers And Different Debt](#) for more information.
- Write off debt class is one component that control how the system writes off an account's stopped service agreements. Refer to [Different Write-Off Criteria For Different Customers And Different Debt](#) for more information.

The severance criteria are used when a collection event is activated that indicates that service should be severed.

SA Type - Billable Charge Template

Open **Admin > Customer > SA Type > Search** and navigate to the **BC Template** page to define the billable charge templates that can be used on service agreements of a given type.

NOTE:

Only billable charges have billable charge templates. Only service agreements that are defined as Billable Charges (in the Special Role on the Details window) may use the grid on this window.

Description of Page

The information in the **Billable Charge Template** collection defines the SA Type's permissible billable charge templates. A billable charge template contains the default bill lines, amounts and distribution codes used to levy a one-off charge. The following fields are required for each template:

Billable Charge Template Specify the billable charge template. Its description is displayed adjacent.

Use As Default Turn on this switch for the template to be defaulted on new billable charges linked to service agreements of this type (if any).

FASTPATH:

For more information about billable charge templates, refer to [Setting Up Billable Charge Templates](#).

Where Used

This information is used to limit the billable charge templates that can be used for a given SA type.

SA Type - Characteristics

To define characteristics common to all service agreements of a given type, open **Admin > Customer > SA Type > Search** and navigate to the **Characteristics** page.

Description of Page

Use the characteristics collection to define a **Characteristic Type**, **Sequence** and **Characteristic Value** common to all service agreements of this type.

NOTE:

You can only choose characteristic types defined as permissible on a SA Type record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

SA Type - Algorithms

Open **Admin > Customer > SA Type > Search** and navigate to the **Algorithm** page to define the algorithms that should be executed for service agreements of a given type.

Description of Page

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (descriptions of all possible events are provided below).
 - Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
-

WARNING:

These algorithms are typically significant processes. The absence of an algorithm may prevent the system from operating correctly.

The following table describes each **System Event** for which you can define algorithms.

System Event	Optional / Required	Description
Bill Completion	Optional	<p>These algorithms are executed whenever a bill is completed for an account that contains a non-canceled service agreement of this type. The following situations necessitate the definition of a completion algorithm on an SA type:</p> <ul style="list-style-type: none"> - As explained under Technical Implementation of A/R Transfer and Technical Implementation of Routing Billable Charges To Service Providers, when a bill is completed, the system needs to set up the data necessary to interface any "master" SA's charges to the service provider and to transfer the receivable balance from the customer to the service provider. The system will only do this if you specify an appropriate algorithm on the master SA types. - As explained under Billing For SAs Covered By The Non-billed Budget, when a bill is completed for accounts that have a non-billed budget SA, the system needs to distribute the non-billed budget's credit balance to the covered SAs. The system will only do this if you specify an appropriate algorithm on the NBB SA types. - As explained under Overpayment Segmentation, when a bill is completed, the system may apply an excess credit from a prior overpayment to an account's service agreements. <p>Note. Algorithms of this type are called for all non- Canceled service agreements, regardless of whether or not they are billed. If your algorithms should only be processed under certain conditions (for example, only process this algorithm for Active service agreements), then it is the responsibility of the algorithm to check the conditions before continuing.</p> <p>Click here to see the algorithm types available for this system event.</p>
Break NBB SA	Optional	<p>These algorithms are executed when a non-billed budget is manually stopped via the non-billed budget maintenance page.</p>

Click [here](#) to see the algorithm types available for this system event.

Note that the Payment Arrangement algorithm and the Break Pay Arrangement algorithm are mutually exclusive.

Break Pay Arrangement	Optional	<p>These algorithms are by executed by severance events when the event is created for payment arrangement SAs. This algorithm should be specified on SA types with a special role of Payment Arrangement to perform special actions that take place when a customer breaks a payment arrangement. Refer to Monitoring Payment Arrangements for more information about breaking payment arrangements.</p> <p>Click here to see the algorithm types available for this system event.</p>
Budget Eligibility	Optional	<p>These algorithms are executed when determining in a service agreement is eligible for budget. Algorithms of this type are only applicable on SA types that are marked as eligible for budget and may be used to override that setting and indicate that the service agreement is not eligible.</p> <p>For example, maybe service agreements in a certain rate are not eligible. Or perhaps service agreements with a given characteristic value are not eligible.</p> <p>Click here to see the algorithm types available for this system event.</p>
Cut Process Rule	Optional	<p>These algorithms are executed to create a cut process for service agreements of this type. Refer to The Big Picture Of Cut Processes for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
FT Freeze	Optional	<p>These algorithms are executed whenever a financial transaction is frozen that is linked to a service agreement of this type. The following situations necessitate the definition of an FT freeze algorithm:</p> <ul style="list-style-type: none">- As explained under Technical Implementation of Routing Consumption To Service Providers, when a master SA's bill segment is frozen, the system must check if there are any service providers who need the bill segment's consumption. If so, it sets up the data necessary to interface the master SA's consumption (snapshot on the bill segment) to the service provider(s). The system will only do this if you specify

an appropriate FT Freeze Algorithm on the master SA types.

- As explained under [Technical Implementation of Paying The Service Provider](#), when a financial transaction (FT) is frozen that is associated with a sub SA, the system must check if this FT should trigger the "payment" of a service provider. If so, it has to create an adjustment to increase how much we owe the service provider. The system will only do this if you specify an appropriate FT Freeze Algorithm on the sub SA types.

Click [here](#) to see the algorithm types available for this system event.

High Bill Amount	Optional	<p>This algorithm type checks for high bill amounts during batch billing. A bill threshold amount can be defined on a service agreement. By default the system compares this threshold amount to the bill segment amount during batch billing. If the bill segment amount exceeds the given threshold value, a bill error is generated. No proration takes place on the threshold amount if the bill segment's consumption period falls outside of the rate frequency's normal period.</p> <p>This algorithm may be used to override the default system behavior. It will prorate the SA's bill threshold amount before comparing it to the batch bill segment amount if the consumption period falls outside of the rate frequency's normal period.</p> <p>Click here to see the algorithm types available for this system event.</p>
Landlord Reversion	Optional	<p>These algorithms are used to create, update or cancel a service agreement for a landlord when service is started, stopped or updated for a premise that references a landlord agreement.</p> <p>Algorithms of this type are called:</p> <ul style="list-style-type: none">- When starting a new SA (start initiation)- When stopping an SA (stop initiation)- When canceling a pending start SA- When changing the stop / start dates of back-to-back service agreements <p>Click here to see the algorithm types available for this system event.</p>
Loan Interest Charge	Optional	<p>These algorithms are executed whenever the interest charge needs to be calculated for a loan, such as when the loan amortization</p>

schedule is created and when a bill segment is created for a loan SA. This algorithm should be specified on SA types with a special role of Loan. Refer to [Defining Loan Options](#) for more information.

Click [here](#) to see the algorithm types available for this system event.

Loan Periods and Amount	Optional	<p>These algorithms are executed whenever a user clicks the Calculate button on Loan - Main or on the Start SA Confirmation dialog for a loan SA. It calculates either the number of periodic payments or the payment amount (depending on whether the user specifies the number of payments or the payment amount as input). This algorithm should be specified on SA types with a special role of Loan. Refer to Defining Loan Options for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Loan Schedule	Optional	<p>These algorithms are executed whenever the system needs to create a loan amortization schedule for a loan, such as when you renegotiate the terms of a loan on Loan - Main. This algorithm should be specified on SA types with a special role of Loan. Refer to Defining Loan Options for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Payment Arrangement	Optional	<p>These algorithms are by executed to handle the creation, breaking and canceling of payment arrangement SAs. This algorithm should be specified on SA types with a special role of Payment Arrangement to perform special actions that take place during the lifecycle of a payment arrangement. Refer to Monitoring Payment Arrangements for more information about payment arrangements.</p> <p>Note that the Payment Arrangement algorithm and the Break Pay Arrangement algorithm are mutually exclusive.</p>
Payment Freeze	Optional	<p>These algorithms are executed whenever a payment is frozen. The following situations necessitate the definition of such an algorithm:</p> <ul style="list-style-type: none">- For a loan SA, such an algorithm is required to create a frozen adjustment that transfers any credit balance resulting from an overpayment to the loan's principal balance. Refer to Defining Loan Options for more information.

		Click here to see the algorithm types available for this system event.
Pre-Bill Completion	Optional	<p>These algorithms are executed immediately prior to bill completion when a bill contains a bill segment for a service agreement whose SA type has such an algorithm. The following situations necessitate the definition of such an algorithm:</p> <ul style="list-style-type: none"> - If you want to delete a bill segment that's in error on the last night of a bill cycle when there are other bill segments that aren't in error, use such an algorithm. <p>Click here to see the algorithm types available for this system event.</p>
Process NBB Scheduled Payment	Optional	<p>These algorithms are executed by the NBB Scheduled Payment Processing background process whenever a scheduled payment is due. If the non-billed budget SA is unmonitored, this algorithm is not called. This algorithm should be specified on non-billed budget SA types to create the necessary adjustments for the non-billed budget SA.</p> <p>Click here to see the algorithm types available for this system event.</p>
Proposal SA Acceptance	Optional	<p>These algorithms are executed when a proposal service agreement is accepted. Refer to Enabling The Creation Of A Real Service Agreement for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Proposal SA Bill Segment Generation	Optional	<p>These algorithms are executed to generate simulated bills segments for a proposal service agreement. Refer to Enabling The Generation Of Simulated Bill Segments for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
Proposal SA Creation	Optional	<p>These algorithms are executed when a proposal service agreement is created. Refer to Enabling The Automatic Generation Of Billing Scenarios for more information.</p> <p>Click here to see the algorithm types available for this system event.</p>
SA Activation	Optional	<p>These algorithms are executed when a service agreement status changes from Pending Start to Active. It performs any additional activities that are necessary to activate an SA. The following situations</p>

necessitate the definition of such an algorithm:

- If you want to create a customer contact to indicate that a non-billed budget has been activated, use such an algorithm.

Click [here](#) to see the algorithm types available for this system event.

SA Cancel

Optional

These algorithms are executed when a service agreement status changes to Canceled. It performs any additional activities that are necessary to cancel an SA.

An example of when you may use this algorithm is that perhaps your business rules dictate that the creation of a payment arrangement should create a credit rating history transaction. When a payment arrangement SA is canceled, the credit rating should be updated with an end date.

Click [here](#) to see the algorithm types available for this system event.

SA Creation

Optional

These algorithms are executed when a service agreement is created. The following situations necessitate the definition of such an algorithm on an SA type:

- If you want to create a To Do entry whenever a new service agreement of a given type is added, specify such an algorithm.
- If you want to automatically activate SAs of a given type (instead of waiting for the background SA activation process to run), specify such an algorithm.
- If you want to create a Workflow Process when a service agreement of a given type is added, specify such an algorithm.

Click [here](#) to see the algorithm types available for this system event.

SA Information

Optional

We use the term "SA information" to describe the basic information that appears throughout the system to describe a service agreement. The data that appears in "SA information" is constructed using this algorithm.

Plug an algorithm into this spot to override the "SA information" algorithm on installation options or the system default "SA information" if no such algorithm is defined on installation options.

Click [here](#) to see the algorithm types available for this system event.

SA Renewal

Optional

These algorithms are executed by the Service Agreement Renewal background process

whenever an SA is due for renewal or when the user clicks the **Renew** button (for **non-billed budgets**). It performs any activities that are necessary to renew an SA and returns the new renewal and expiration dates for the SA.

Click [here](#) to see the algorithm types available for this system event.

SA Stop

Optional

These algorithms are executed whenever a service agreement's status changes from pending stop to stopped. The following situations necessitate the definition of a SA stop algorithm:

- For **non-billed budgets** to distribute any remaining credit balance from the non-billed budget SA to the covered SAs, you must specify such an algorithm.

- For service credit memberships that have a **refundable membership fee**, an SA Stop algorithm attempts to refund the fee if this is the last SA linked to the membership that is being stopped.

Click [here](#) to see the algorithm types available for this system event.

SA Stop Initiation

Optional

These algorithms are executed whenever a service agreement's status becomes pending stop. The following situations necessitate the definition of a stop initiation algorithm on an SA type:

- As explained under **Finalizing Pending Stops**, service agreements are normally transitioned from pending stop to stopped by a background process (or manually). For **non-billed budget SAs** to transition to stopped automatically (without waiting for the background process), you must specify such an algorithm.

When does a SA become pending stop?

Service agreements typically become pending stop when a user initiates a request to stop service on **Start Stop - Main**. A **severance process** with an Expire SA **severance event** causes a service agreement to become pending stop (when the event is executed). Additionally, the **Stop Expired Service Agreements** background process starts the process to initiate the stop of an SA when the expiration date is on or before the process date.

Click [here](#) to see the algorithm types available for this system event.

Start Stop Field Work

Optional

These algorithms are executed to create the field activities necessary to start and stop service. Refer to [Starting Service and Field Activities](#) and to [Stopping Service and Field Activities](#) for a description of when algorithms of this type are called. The following situations necessitate the definition of a start stop fieldwork creation algorithm:

- If a service agreement has field activities created to start and stop service at its service points, its SA type must have an appropriate start stop field work creation algorithm.

Click [here](#) to see the algorithm types available for this system event.

SA Type - Billable Charge Overrides

The [BCU2 - Create Billable Charge](#) background process is responsible for creating billable charges for each billable charge upload staging record interfaced into the system. This process will override the values of the various switches referenced on a billable charge upload staging line if the respective service agreement's SA type has an override value for the billable charge upload staging line's billable charge line type.

NOTE:

This information is optional. If you don't need to override the values of a [Billable Charge Line Type](#) you don't need to set up this information.

Open **Admin > Customer > SA Type > Search** and navigate to the **BC Upload Override** page to define override values for a given SA Type / Billable Charge Upload Staging Line Type.

Description of Page

Use the **Billable Charge Overrides** collection to define values to be overridden on billable charge lines uploaded from an external system (refer to the description above for the details). The following switches may be overridden for a given **SA Type** and **Billable Charge Line Type**.

- Use the **Show on Bill** switch to define the value to be defaulted into the Show on Bill indicator on billable charge upload lines that reference this line type.
- Use the **Appear in Summary** switch to define the value to be defaulted into the App in Summary indicator on billable charge upload lines that reference this line type.
- Use **Memo Only, No GL** switch to define the value to be defaulted into the Memo Only, No GL indicator on billable charge upload lines that reference this line type.
- Use **Distribution Code** to define the value to be defaulted into the Distribution Code on billable charge upload lines that reference this line type.

SA Type - Contract Option Type

Open **Admin > Customer > SA Type > Search** and navigate to the **Contract Option Type** page to define the contract option types, which are valid for service agreements of a given type.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

FASTPATH:

Contract Options are used by the system to define special options under which certain calculations supporting a contract's rate may be overridden or altered occasionally for specific periods of time. Refer to [Contract Option Background Topics](#) for more information.

Description of Page

If the SA Type's special role is Interval, you may define the **Contract Option Types** that are valid for contract options linked to a service agreement of this type.

FASTPATH:

For more information, refer to [Setting Up Contract Option Types](#).

Where Used

This information is used to validate the contract options linked to the service agreement. Refer to [Service Agreement - Contract Options](#) for more information.

SA Type - Interval Info

Open **Admin > Customer > SA Type > Search** and navigate to the **Interval Info** page to define the interval profile relationship types and TOU map relationship types, which are valid for service agreements of a given type.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

Description of Page

If the SA Type's special role is Interval , you may define the **Interval Profile Relationship Types** that may be linked to service agreements of this type.

FASTPATH:

For more information about interval profile relationship types, see [Designing Interval Profile Relationship Types](#).

If the SA Type's special role is Interval , you may define the **TOU Map Relationship Types** that may be linked to service agreements of this type.

FASTPATH:

For more information about TOU map relationship types, refer to [Designing TOU Map Relationship Types](#).

Where Used

The interval profile information is used to validate the interval profile relationship types linked to the service agreement. Refer to [Service Agreement - Interval Info](#) for more information.

The TOU map information is used to validate TOU map relationship types linked to a service agreement. Refer to [Service Agreement - Interval Info](#) for more information.

SA Type - NBB Recommendation Rule

Open **Admin** > **Customer** > **SA Type** > **Search** and navigate to the **NBB Rec'n Rule** page to define the recommendation rules that are valid for non-billed budget SAs of this type.

Description of Page

If the SA Type's special role is Non-billed Budget, you may define the **Recommendation Rules** that are valid on non-billed budget SAs of this type. Check the **Use As Default** box to indicate the default recommendation rule for service agreements of this type.

FASTPATH:

For more information about non-billed budgets, refer to [Defining Non-Billed Budget Options](#).

Where Used

The non-billed budget recommendation rules are used to recommend the payment amount and payment schedule for non-billed budget service agreements. Refer to [Maintaining Non-billed Budgets](#) for more information.

Setting Up Start Options

Start options save users time and prevent data entry errors because they default many values on a service agreement (e.g., the rate schedule, recurring charge amount, contract riders, contract terms, characteristics, terms and conditions, etc. can all be defaulted onto a service agreement from a start option).

A SA type may have zero or more start options.

- A SA type without start options is usually one that has a very limited number of options. For example, if a SA type has a single valid rate and no customer-specific contract values, you don't need to setup a start option (the SA's default rate can default based on the information defined when you setup the SA type).
- A SA type with multiple start options is one where many different permutations are possible. For example, a SA type that can have multiple rates and each rate can have multiple riders is a good candidate for start options (where each start option will default, for example, a specific rate and set of contract riders).

When the [Start/Stop](#) transaction is used to start service AND the service being started uses a SA type with start options, the user is asked to select one of the start options. The service agreement that's created is populated with fields from the start option.

When the [Order](#) transaction is used to start service, the user selects start options, but only indirectly. It works like this:

- A user selects a "package" of services to start service for a customer.
- A "package" contains one or more start options.
- When a user selects a package, the system creates a service agreement for each start option on the package (and defaults the information on each service agreement from the respective start option).

A start option's default values may change over time (i.e., the information on a start option is effective-dated). The start service logic uses the version of the start option that is effective on the day service starts.

Start options can cause a great deal of information to be populated on a service agreement. There are several ways to change this default information:

- A user may override this information using the [Service Agreement](#) transaction.
- If the service agreement is in the pending start state, you can use the [Start/Stop - Pending SAs](#) page to change the service agreement's SA type and/or start option.

- If the service agreement is in active or pending stop states, a button appears on [Service Agreement - Main](#) called **Apply New Start Option**. When pressed, the user is allowed to define a start option and the date its terms become effective on the service agreement. Refer to [Changing A Start Option](#) for the details of this functionality.

The topics in this section describe how to setup start options.

NOTE:

The merge transaction can save setup time. The [Start Options Merge](#) transaction can be used to construct a start option by copying pieces from other start options.

Start Option Considerations - Rate-Oriented SA Types

To understand the following discussion, you should be familiar with the following concepts:

- SA types that use rates have one or more valid rate schedules. Only these rate schedules may be defined on service agreements of a given type.
- One of a SA type's rate schedules may be designated as the "default" rate. The system assigns the "default" rate to new service agreements when a CSR doesn't choose a start option.
- Start options may be used by customer service reps when a service is started for a customer. A start option causes the customer's new service agreement to be populated with a specific rate and contract terms (e.g., contract riders, contract values). The use of start options is not allowed if the service's SA type does not require a start option. The use of start options is required if the service's SA type requires a start option.

Whether or not a SA Type has start options is dependent on the following factors:

- If a SA Type has only one valid rate and the rate doesn't use customer-specific contract terms (e.g., contract riders, contract values), the SA Type does not need any start options. Why? Because the system default's the SA type's default rate on new service agreement when no start option is used at start time.
- If a SA Type has only one valid rate, but under unusual circumstances, it uses customer-specific contract terms, you'll want an option for every situation (both the standard one and the unusual ones).
- If a SA Type has a single rate with a variety of contract terms (which, by definition, are different for each customer), then you'd want a start option for each permissible combination of contract terms. You'd also want to turn on the SA type's Start Option Required switch to make your CSR's pick one of the start options when service is started (rather than let the system use the SA type's default rate).
- If a SA type has multiple valid rates and a variety of contract terms (a combination of the previous two points), you'd need a start option for each permissible combination. You'd also want to turn on the SA type's Start Option Required switch to make your CSR's pick one of the start options when service is started (rather than let the system use the SA type's default rate).

Start Option Considerations - SA Types That Use Recurring Charges

To understand the following discussion, you should be familiar with the following concepts:

- Many SA types use the recurring charge algorithms to generate the bill segments. For example, if you let a customer pay for a \$900 heat pump in 3 installments of \$300, you'd have an SA type called CA/MERCH-I and indicate it uses the recurring charge algorithm.
- When a CSR creates an SA type that uses recurring charge algorithms, they can enter the recurring charge amounts on [Start/Stop Service - Start Confirmation](#) window OR they can specify a start option on this window and let the system populate the recurring charge amount. For example, if you let a customer pay for a \$900 heat pump in 3 installments of \$300, you can set up a start option called HP 3PAY with an installment amount of \$300.

- Depending on the type of service being started, you might also need to generate an adjustment when service is started in order to initialize the total debt. For example, when a customer buys a heat pump we'll need an adjustment issued to realize the entire \$900 of revenue.

The following table provides examples of recurring charge SA types with several typical start options:

CIS Division/ SA Type	Start Option	Adjustment Type	Install Amount	Comments
CA/CHARITY	DONATE \$5		5	This causes a charitable contribution to be created with an installment amount of \$5.
CA/CHARITY	DONATE \$10		10	This causes a charitable contribution to be created with an installment amount of \$10.
CA/PA-REGU	PAY \$10 PM		10	This causes a payment arrangement to be created with an installment amount of \$10.
CA/PA-REGU	PAY \$20 PM		20	This causes a payment arrangement to be created with an installment amount of \$20.
CA/PA-UNRE	PAY \$10 PM		10	This causes a payment arrangement to be created with an installment amount of \$10.
CA/PA-UNRE	PAY \$20 PM		20	This causes a payment arrangement to be created with an installment amount of \$20.
CA/MERCH-I	HP 3PAY	HEATPUMP	300	This causes a merchandise service agreement to be created with an installment amount of \$300. It also causes an adjustment to be issued to realize the \$900 of revenue.
CA/MERCH-I	BBQ 3PAY	BBQ	250	This causes a merchandise service agreement to be created with an installment amount of \$250. It also causes an adjustment to

			be issued to realize the \$750 of revenue.
CA/DEP-I	PAY \$40 PM	40	This causes a deposit service agreement to be created with an installment amount of \$40. The CSR would be required to define the total deposit amount to be billed over the life of the service agreement on the secondary SA window on the Start Account window.

Start Option Considerations - Initial Adjustment SA Types

To understand the following discussion, you should be familiar with the following concepts:

- Some SA types depend on an adjustment to book their initial debt. For example, the CA/MERCH-I SA type requires an initial adjustment to book the payoff amount for the entire cost of the merchandise. Another example would be a loan service agreement (refer to [Booking The Principal Amount Using An Adjustment](#) for the details).
- When a CSR starts an SA type that requires an initial adjustment, they can create the adjustment immediately after starting service OR then can specify a start option when they start service and let the system generate the adjustment. You could let the system create the adjustment to book the \$900 associated with the heat pump as shown in the previous section.

The following table provides examples of SA types that use adjustments with several typical start options:

CIS Division/ SA Type	Start Option	Adjustment Type	Install Amount	Comments
CA/MERCH-I	HP 3PAY	HEATPUMP	300	This causes a merchandise service agreement to be created with an installment amount of \$300. It also causes an adjustment to be issued to realize the \$900 of revenue.
CA/MERCH-I	BBQ 3PAY	BBQ	250	This causes a merchandise service agreement to be created with an installment amount of \$250. It also causes an adjustment to be issued to realize the \$750 of revenue.
CA/CONNECT	CONNECT	CONNECT		This causes an adjustment to be issued

Start Option Considerations - Interval SA Types

Refer to [Designing Your SA Interval Billing Options](#) for information about setting up start options for interval service agreements.

Start Option - Main

Open **Admin > Customer > SA Type Start Option > Add** to define a SA type's start options.

Description of Page

Every start option is uniquely identified by the following fields:

CIS Division & SA Type Enter the Division and SA type to which the start option is linked.

Start Option Enter the unique identifier of the option. Pick something easy to recognize as this will be used by CSRs to pick an option when they start service.

Effective Date Enter the earliest effective date. It should be the same as the earliest effective date of the start option's rate (although it doesn't hurt for it to be earlier). The date defaults to the current date. (The status, below, should be Active.)

The remaining fields further describe a start option.

Enter a **Description** for the start option.

Indicate its **Status**. For new start options, the status should be Active. When it's no longer applicable, change it to Inactive.

Enter the primary **Rate Schedule** that should be defaulted onto service agreements created using this option. Refer to [Start Option Considerations For SA Types That Use Rates](#) for more information.

FASTPATH:

For more information about a service agreement's rates, refer to [Service Agreement - Rate Info](#).

Enter the **Adjustment Type** that should be generated, if any, when service is started using this option. Refer to [Start Option Considerations For SA Types That Use Initial Adjustments](#) for more information.

Enter the **Recurring Charge Amount** that should be defaulted onto service agreements created using this option. This field is only visible when the SA type allows recurring charges. In addition, the prompt for this field is defined on the SA type table on the billing window (e.g., it could appear as Payment Amount, Budget Amount, or Installment Amount). Refer to [Start Option Considerations For SA Types That Use Recurring Charges](#) for more information.

Enter the **Currency Code** in which monetary amounts are denominated.

NOTE:

The currency code defaults from the [installation record](#).

Enter the **Total Amount to Bill** that should be defaulted onto service agreements created using this option. This is useful to initiate either a loan or a deposit. The prompt for this field is defined on the SA type table on the billing window (e.g., it could be Deposit Amount or Loan Amount).

Use **Number of Payment Periods** to default the number of payment periods of service agreements created using this option. This field is only allowed for SA types with special role of Loan. Refer to [The Terms Of A Loan Are Stored On A Service Agreement](#) for more information.

If your SA Type has a special role of Interval, then you must enter the **Cutoff Time** and **Start Day Option** that should be defaulted onto service agreements created using this option. For start day option, you may choose Current Day or Previous Day. Refer to [Start and End Times for Billing](#) for more information about how these fields are used.

If your SA Type has a special role of Billable Charge, then you can setup the start option to automatically create a billable charge when a service agreement is created using this start option. For example, you might have a start option that automatically creates a "one-time invoice" service agreement along with a "tree trimming" billable charge. To use this feature you should turn on the **Create Billable Charge** switch and specify the **Billable Charge Template** that will be used to create the billable charge. These fields are only allowed for SA types with special role of Billable Charge.

FASTPATH:

Refer to [Setting Up Billable Charge Templates](#) for more information about templates. Refer to [An Easier Way To Create One Time Charges](#) for an example of how you can setup a campaign with packages that use this type of start option.

NOTE:

The duplicate action in the page actions toolbar enables you to copy another start option. Refer to [Duplicate Button](#) in the system wide standards document for more information.

Start Option - Rate Info

Open **Admin > Customer > SA Type Start Option > Search** and navigate to the **Rate Info** page to define the start option's default values for contract riders and contract values.

Description of Page

The information in the **Contract Riders** collection defines the contract riders to be defaulted onto service agreements created using this start option. The following fields are required for each instance:

Bill Factor The bill factor defines the type of rider. You may only reference bill factors designated as being applicable for contract riders.

Number of Days The number of days the rider should be in effect. This value is used by the system to set the stop date on the service agreement's contract rider. If the rider has no expiration, set this field to 0. Default note: this field will be set to 0 if left blank.

FASTPATH:

For more information about a rate's contract riders, refer to [Defining General Bill Factor Information](#). For more information about a service agreement's contract riders, refer to [Service Agreement - Contract Riders](#).

The information in the **Contract Values** collection defines the contract values to be defaulted onto service agreements created using this start option. The following fields are required for each event:

Bill Factor The bill factor defines the type of value. You may only reference bill factors designated as allowing values in contract terms.

Number of Days The number of days the value should be in effect. This value is used by the system to set the stop date on the service agreement's contract value. If the value has no expiration, set this field to 0.

Value The amount of the contract value.

FASTPATH:

For more information about a rate's contract values, refer to [Defining General Bill Factor Information](#). For more information about a service agreement's contract values, refer to [Service Agreement - Contract Values](#).

Start Option - Characteristics & Qty

Open **Admin > Customer > SA Type Start Option > Search** and navigate to the **Characteristics & Qty** page to define the start option's default values for characteristics and contract quantities.

Description of Page

The information in the **Characteristics** collection defines the characteristics to be defaulted onto service agreements created using this start option. The following fields are required for each instance:

Characteristic Type This defines the type of characteristic. Note: you may only define characteristics valid on service agreements.

Characteristic Value This defines the characteristic value that will be defaulted.

FASTPATH:

For more information about a service agreement's characteristics, refer to [Service Agreement - Characteristics](#).

The information in the **Contract Quantity** collection defines the contract quantities to be defaulted onto service agreements created using this start option. The following fields are required for each instance:

Contract Quantity Type This defines the type of contract quantity.

Contract Quantity The amount of the contract quantity.

FASTPATH:

For more information about a service agreement's contract quantities, refer to [Service Agreement - Contract Quantity](#).

Start Option - Contract Option

FASTPATH:

Refer to [Designing Your SA Interval Billing Options](#) for information about setting up start options for interval service agreements.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

Open **Admin > SA Type Start Option > Search** and navigate to the **Contract Option** page to define the start option's contract option default values.

Description of Page

The collection of contract option types will be used to link contract options to a service agreement that will be created with this start option. If the contract option to create for the service agreement will be SA Specific, simply indicate the **Contract Option Type**. This will cause a new contract option to be created with this contract option type and the new contract option will be linked to the new SA at start time. If the contract option to create for the service agreement will be Shared, indicate the contract option type and the **Contract Option ID**. This will cause the specified contract option to be linked to the new SA at start time.

FASTPATH:

For more information about contract option types, see [Designing Your Contract Option Types](#).

Start Option - Interval Info

FASTPATH:

Refer to [Designing Your SA Interval Billing Options](#) for information about setting up start options for interval service agreements.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

Open **Admin > SA Type Start Option > Search** and navigate to the **Interval Info** page to define the start option's interval profile and TOU map default values.

Description of Page

The collection of interval profile information will be used to link interval profiles to a service agreement that will be created with this start option. The **Interval Profile Relationship Type** indicates the value that will be linked to each new SA Profile record. For each interval profile relationship type, define either an **Interval Profile Type** or an **Interval Profile ID** depending on whether the Profile to be linked to the new service agreement should be SA Specific or Shared. For SA Specific, indicate an interval profile type. This will cause a new interval profile to be created with this interval profile type and the new interval profile will be linked to the new SA at start time. For Shared profiles, indicate the Interval Profile ID. This will cause the specified Profile to be linked to the new SA at start time.

If a **Derivation Algorithm** is linked to the interval profile type, then the algorithm and its **Creation Priority** are displayed.

FASTPATH:

For more information about interval profile relationship types, see [Designing Interval Profile Relationship Types](#). For more information about Shared profiles vs. SA Specific profiles, see [Common Profiles vs. SA Owned Profiles](#).

The collection of TOU map information will be used to link TOU Maps to a service agreement that will be created with this start option. The **TOU Relationship Type** indicates the value that will be linked to each new SA TOU Map record. For each TOU relationship type, define either a **TOU Map Type** or a **TOU Map ID** depending on whether the map to be linked to the new service agreement should be SA Specific or Shared. For SA Specific, indicate a TOU map type. This will cause a new TOU map to be created with this TOU map type and the new TOU map will be linked to the new SA at start time. For Shared map, indicate the TOU map id. This will cause the specified map to be linked to the new SA at start time.

If a TOU map creation algorithm is linked to the TOU Map type, then the **Derivation Algorithm** and its **Creation Priority** are displayed.

FASTPATH:

For more information about TOU map relationship types, see [Designing Your TOU Map Relationship Types](#).

Start Option - TOU Contract Value

FASTPATH:

Refer to [Designing Your SA Interval Billing Options](#) for information about setting up start options for interval service agreements.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

Open **Admin > SA Type Start Option > Search** and navigate to the **TOU Contract Value** page to define the start option's default TOU contract values.

Description of Page

The **TOU Contract Value** scroll contains information to define when you would like TOU contract values to be defaulted onto service agreements created using this start option

Indicate the **Bill Factor** associated with the TOU contract values. You may only reference bill factors designated as "TOU" bill factor types and where the value may be in contract terms.

Indicate the **TOU Group**, which contains the appropriate collection of TOU codes for service agreements created using this start option.

Indicate the **Number of Days** the value should be in effect. This value is used by the system to set the stop date on the service agreement's contract value. If the value has no expiration, set this field to 0.

Use the Time of Use collection to indicate the **TOU Value** associated with each **Time of Use** codes for the TOU Group.

FASTPATH:

For more information about TOU contract values, refer to [Customer Specific TOU Values](#).

Start Option - Terms and Conditions

Open **Admin > Customer > SA Type Start Option > Search** and navigate to the **Terms and Conditions** page to define the start option's default terms and conditions.

Description of Page

The information in the grid defines the terms and conditions to be defaulted onto service agreements created using this start option. The following fields are required for each instance:

Terms and Conditions This is the code that identifies a term and condition (T&C).

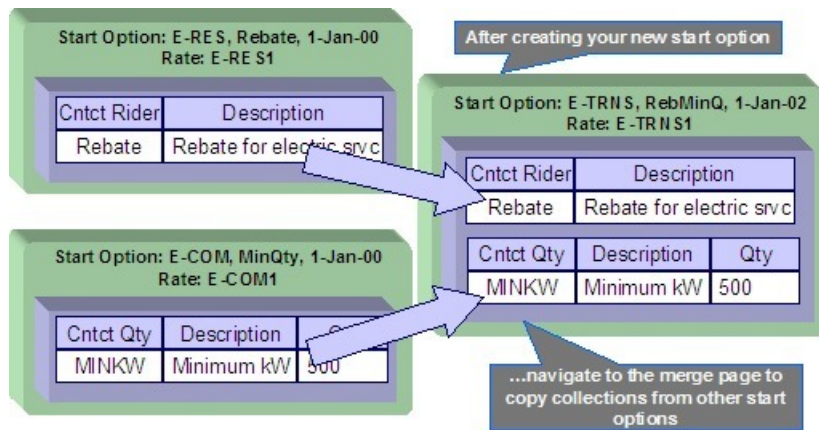
Number of Days The number of days the T&C should be in effect. This value is used by the system to set the end date on the service agreement's T&C. If the T&C has no expiration, set this field to 0. Default note: this field is set to 0 if left blank.

Start Option Merge

Use this page to modify an existing start option by copying information from other start options. This page may be used to copy records from the contract rider, contract value, contract quantity, characteristic, interval profile, TOU map, contract option and TOU contract value collections from one or more existing start options to another.

NOTE:

The target start option must exist prior to using this page. If you are creating a new start option, you must first go to the [Start Option](#) page to add the new start option and then navigate to the merge page to copy collection information.



NOTE:

Duplicate versus Merge. The **Start Option** page has **Duplication** capability. You would duplicate a start option if you want to a) create a new start option AND b) populate it with all the information from an existing start option. You would use the start option merge page if you want to build a start option using pieces of one or more start options.

Start Option Merge - Main

Open **Admin > Customer > SA Type Start Option Merge.**

Description of Page

Select the **Original Start Option**, which is the target for merging the start option collection information.

Select the **Merge From Start Option**, which is your template start option to copy the collections from.

NOTE:

You may only copy information from one Merge From start option at a time. If you wish to copy information from more than one start option, select the first Merge From start option, copy the desired records, Save, then select the next Merge From start option.

The left portion of the page will display any existing records in the collections for the original start option. The right portion of the page will display the existing records in the collections for the Merge From start option.

You may use the **Copy All** button to copy all the records in all the collections from the Merge From start option to the Original start option. If you do not choose to copy all, you may copy records individually as described below.

The left portion of the **Contract Riders** collection initially displays existing contract riders linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Bill Factor** and **Number of Days** for each contract rider is displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The left portion of the **Contract Values** collection initially displays existing contract values linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Bill Factor**, **Number of Days** and **Value** for each contract value is displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The topics, which follow, describe how to perform common maintenance tasks:

Removing A Row From A Grid

If you wish to remove a record linked to the Original start option, click the "-" button to the left of the record.

Adding A New Row To A Start Option

You may move any of the records from the Merge From start option to the original start option by selecting the left arrow adjacent to the desired row. Once a record is moved it will disappear from the Merge From information and appear in the Original information with the word Merge in the Merge Type column.

Removing An Uncommitted Row From A Start Option

If you have copied a row across by mistake, you may remove it by clicking on the right arrow adjacent to the appropriate record.

FASTPATH:

Refer to [Editable Grid](#) in the system wide standards documentation for more information about adding records to a collection by selecting from a list.

Start Option Merge - Characteristics and Quantities

Open **Admin > Customer > SA Type Start Option Merge** and navigate to the **Characteristics and Quantities** page to copy rows in the characteristic and contract quantity collections.

Description of Page

The left portion of the **Characteristics** collection initially displays existing characteristics linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Characteristic Type** and **Characteristic Value** for each characteristic are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The left portion of the **Contract Quantity** collection initially displays existing contract quantities linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Contract Quantity Type** and **Contract Quantity** for each contract quantity are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

FASTPATH:

Refer to [Start Option Merge - Main](#) for more information about how to perform common maintenance tasks for the grids displayed on this tab page.

Start Option Merge - Contract Option

Open **Admin > Customer > SA Type Start Option Merge** and navigate to the **Contract Option** page to copy rows in the contract options collection.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

Description of Page

The left portion of the **Contract Option** collection initially displays existing contract option information linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Contract Option Type**, **Contract Option ID** and **Description** of the contract option for each contract option row are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

FASTPATH:

Refer to [Start Option Merge - Main](#) for more information about how to perform common maintenance tasks for the grids displayed on this tab page.

Start Option Merge - Interval Info

Open **Admin > Customer > SA Type Start Option Merge** and navigate to the **Interval Info** page to copy rows in the interval profile, and TOU map and TOU contract quantity collections.

NOTE:

This tab may not appear. This tab is suppressed if the interval billing Complex Billing module is [turned off](#).

Description of Page

The left portion of the **Interval Profiles** collection initially displays existing interval profile information linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Interval Profile Relationship Type**, **Interval Profile Type** and **Interval Profile ID** for each interval profile row are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The left portion of the **TOU Maps** collection initially displays existing TOU map information linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **TOU Map Relationship Type**, **TOU Map Type** and **TOU Map ID** for each TOU map row are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The left portion of the **TOU Contract Values** collection initially displays existing TOU contract values linked to the original start option. In the **Merge Type**, you will see the word Original, for any of these records. The **Bill Factor** and **TOU Group** for each TOU contract value are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

FASTPATH:

Refer to [Start Option Merge - Main](#) for more information about how to perform common maintenance tasks for the grids displayed on this tab page.

Start Option Merge - Terms and Conditions

Open **Admin > Customer > SA Type Start Option Merge** and navigate to the **Terms and Conditions** page to copy terms and conditions (T&Cs).

Description of Page

The left side of the **Terms and Conditions** grid initially displays the T&Cs linked to the original start option. On the right side, the T&Cs linked to the merge from start option are displayed initially.

FASTPATH:

Refer to [Start Option Merge - Main](#) for a description of how to perform common maintenance tasks for the grids displayed on this tab page.

Defining SA Relationship Options

We use the term "SA Relationship" to describe functionality that supports the following situations:

- When companies other than your own provide a service to your customers AND you have some type of interaction with these companies. For example, in a deregulated market, customers deal with both distribution and energy supply companies. These companies typically exchange a great deal of information about their joint customers.
- When multiple rates are associated with a service (where each rate corresponds with a sub-category of service). For example, most water companies charge for both water and wastewater service using separate rates for each. While it is possible to set up water and wastewater as separate service agreements, the SA relationship functionality allows you to set up a single "master" service agreement (for the water service) and associate with it a "sub" service agreement (for the wastewater service).
- When a party representing a group of customers negotiates a contract that is applied over and above those of the individual service agreements. For example, the head office of a national chain may negotiate for additional discounts that should be calculated together or individually. The SA relationship functionality may be used to track the covered service agreements and to calculate and transfer discounts to the head office's service agreement. Refer to the special discounts section.

WARNING:

Setting up the SA relationship control tables is as challenging as your organization's business rules. If you don't have requirements similar to those described above, you don't have to set up anything. If you have these types of requirements, your setup process will be taxing as you must design and set up control tables that manage the financial and consumption interactions that take place between you, your customers, and the various service providers.

The topics in this section describe tables that control your SA relationship functionality. Refer to [Defining Workflow and Notification Options](#) for a description of the tables that control how your organization communicates with the service providers who provide service to your customers.

The Big Picture of SA Relationships and Service Providers

You must set up service providers if companies other than your own provide a service to your customers AND you have some type of interaction with these companies. You will have one service provider for each such company. For example, if you are a distribution company in a deregulated market, you will have a service provider for each company that provides any of the following services:

- Energy (commonly referred to as Energy Supply Companies, Energy Service Providers, Retailers, and Suppliers)
- Meter service (commonly referred to as Meter Service Providers and Meter Agents)
- Meter reading (commonly referred to as Meter Data Management Agencies and Meter Reading Service Providers)
- Billing (commonly referred to as Billing Agents)

The topics in this section provide background information about service providers.

Persons and Service Providers

A great deal of information about your service providers is defined using a person. For example, a service provider's name, address, phone numbers, electronic ID's, etc. are all defined on the person object.

In addition, every service provider must have a service provider object created for it. The service provider object contains information about a provider's relationships with the customer and your organization, for example:

- Do you calculate bills for the service provider? If so, do you have their rates or do they interface their charges to you? Refer to [Billing Relationships](#) for more information.
- Do you send the customers' consumption to the service provider? Do they send it to you? Refer to [Consumption Relationships](#) for more information.
- How are financial settlements between your organization and the service provider implemented (do you pay them when you get paid?, do you purchase the receivable for them?, etc.). Refer to [Service Providers Have Service Agreements Too](#) for more information.
- Etc.

FASTPATH:

Refer to [Designing Your SA Relationship Types and Service Providers](#) for more information.

NOTE:

In some situations, you will need to set up a service provider for your own company. Refer to [When Your Company Is A Service Provider](#) for the details.

Service Providers Are Linked To Service Agreements

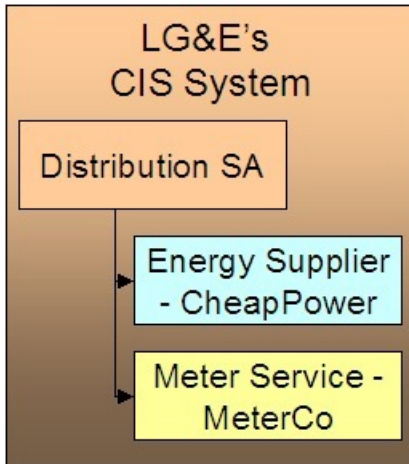
The following diagram illustrates a customer's bill for electric service in a deregulated market. Notice that there are separate sections for energy, distribution and meter service.

NOTE:

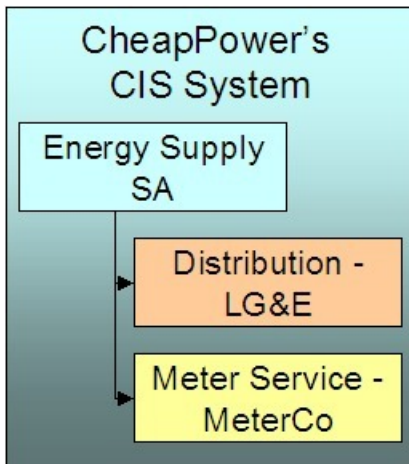
Consolidated billing. The following is an example of a bill that consolidates charges from many service providers. Rather than receive a consolidated bill, it is possible for the customer to receive 3 separate bills, one from each service provider (we refer to this as Dual billing).

Customer's Electric Bill	
Energy Supply - Cheap Power	\$12
Distribution - LG&E	\$9
Meter Service - MeterCo	\$2

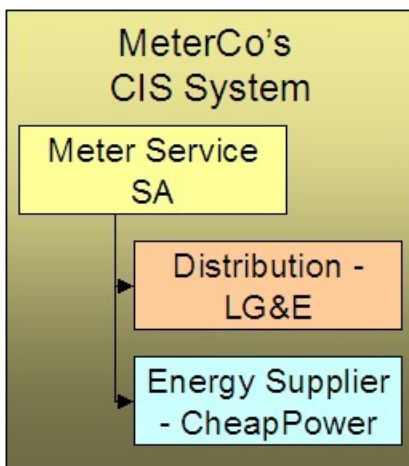
If we were to look at this customer's service agreement in the distribution company's CIS system, we'd find a service agreement for distribution charges, and linked to it would be information about the meter service and energy service providers:



If we look at this customer's service agreement in the energy supply company's CIS system, we'd find a service agreement for energy charges, and linked to it would be information about the distribution and meter service providers:



And finally, if we look at this customer's service agreement in the meter service company's CIS system, we'd find a service agreement for meter service charges, and linked to it would be information about the distribution and energy providers:



NOTE:

Bottom line. A customer's service providers keep track of the customer and each other in their respective CIS systems. A customer will have a service agreement (or the equivalent) in each service provider's CIS system. A customer's service agreement defines the service providers who supply each type of service.

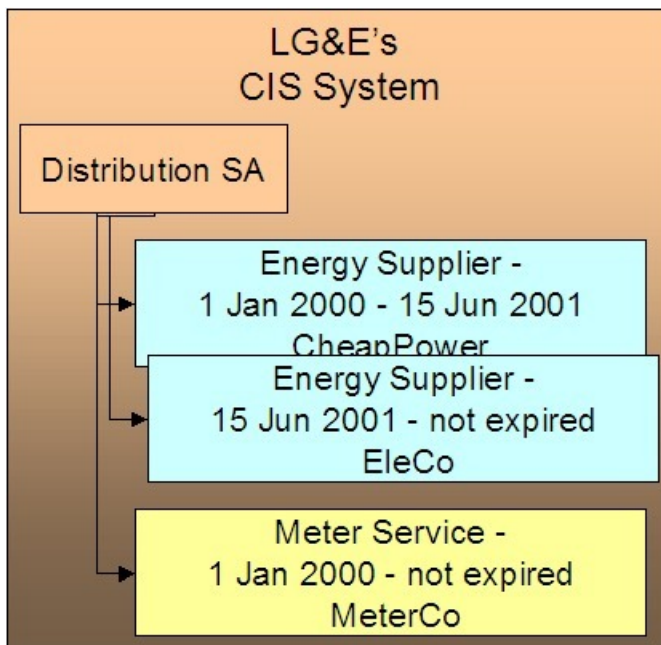
FASTPATH:

Because information about customers and their services needs to be kept up-to-date in many different CIS systems, there is a need for automated communications between service providers. Refer to [Service Providers Have To Communicate About Customers](#) for more information.

Service Providers May Change Over Time

A customer typically has a choice of service providers. Over time, their choice may change. The system keeps track of a customer's service providers throughout time so that it can accurately cancel / rebill historical bills. This means that the service provider relationship is *effective-dated*.

The following diagram illustrates how a distribution company's system keeps track of a customer's service providers (notice the customer changed energy suppliers):



How To Set Up SA Relationships On A Customer's Service Agreement

There are three ways to set up a customer's service providers:

- **Manually.** An operator can manually change a service agreement's SA relationships. The manual method is NOT recommended as changing service providers typically involves many events (e.g., you have to notify the current service provider that they will be dropped). We strongly recommend having an operator kick off a workflow process and let the workflow process notify the service providers and make the desired changes. Refer to [Defining Workflow and Notification Options](#) for more information.

- **Use Default Service Providers.** The system will default a service provider on a required SA relationship when a service agreement is activated. Refer to [Defaulting Relationship Types And Defaulting Service Providers](#) for more information.
- **Workflow Processing.** A workflow process may contain workflow events that change a service agreement's SA relationships. Refer to [Defining Workflow and Notification Options](#) for more information.

When Your Company Is A Service Provider

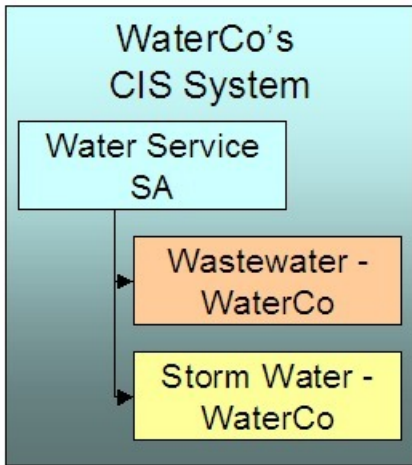
Besides setting up service providers for other companies that provide services to your customer, you may have to set up a service provider for your own company. You have to do this when:

- **Your organization can supply one of the services.** Refer to the illustration in [Service Providers Are Linked To Service Agreements](#). If you are LG&E and you supply energy in addition to distribution, you would need to set up a service provider for your own organization. Why? Because whenever you have a subcategory of service (e.g., energy supply), you must indicate the service provider who provides this service; even when it's you.
- **You decide to break up a service into subcategories** (and have a separate service agreement for each category). For example, a water company may choose to break up service charges into water, wastewater and storm water (they may do this because there are different rates for each category of service). The following is an example of the segregated charges associated with this water company's service charges.

Customer's Water Bill	
Water Service - WaterCo	\$15
Wastewater - WaterCo	\$6
Storm Water - WaterCo	\$5

This water company system may benefit by creating a single service (for water) and indicating there are subcategories of service (for wastewater and storm water). Whenever you have a subcategory of service, you must indicate the service provider who provides this service. And, in this example, the water company would be the sole service provider for each subcategory of service.

Customer's Water Bill	
Water Service - WaterCo	\$15
Wastewater - WaterCo	\$6
Storm Water - WaterCo	\$5

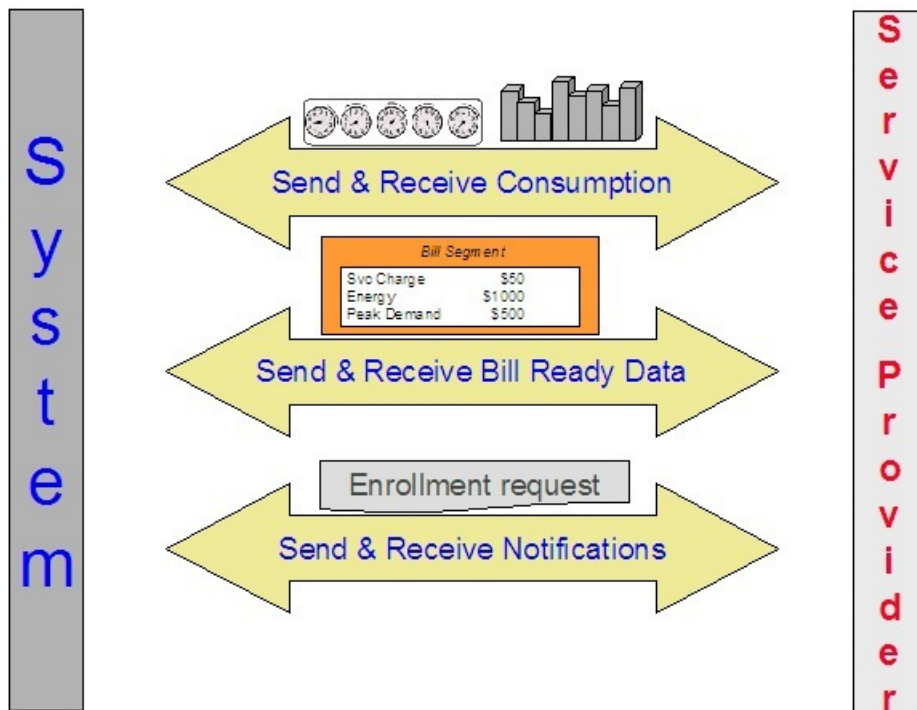


NOTE:

Refer to [We Bill For Them - Rate Ready Bill Segments Are Special](#) for restrictions in respect of using subcategories of service.

Service Providers Have To Communicate About Customers

The providers of service typically have to communicate with each other in respect of the customer's service. The following diagram illustrates the major interfaces of information between your system and your service providers.



Depending on where your organization fits in the service provider hierarchy, you may:

- Bill for other service providers (or they may bill for you). Refer to [Billing Relationships](#) for more information about billing communications.
- Send the customer's consumption to service providers (or they may send it to you). Refer to [Consumption Relationships](#) for more information about consumption communications.
- Apprise service providers of the changes to the customers' service (or they may apprise you). Refer to [How Do You Communicate With Service Providers?](#) for more information about communications between service providers.

Relationships Between Service Providers

Service providers may arrange contractual relationships with other service providers to provide additional services, e.g., an energy service provider may work with a specific meter data management agency to gather and report interval meter read data.

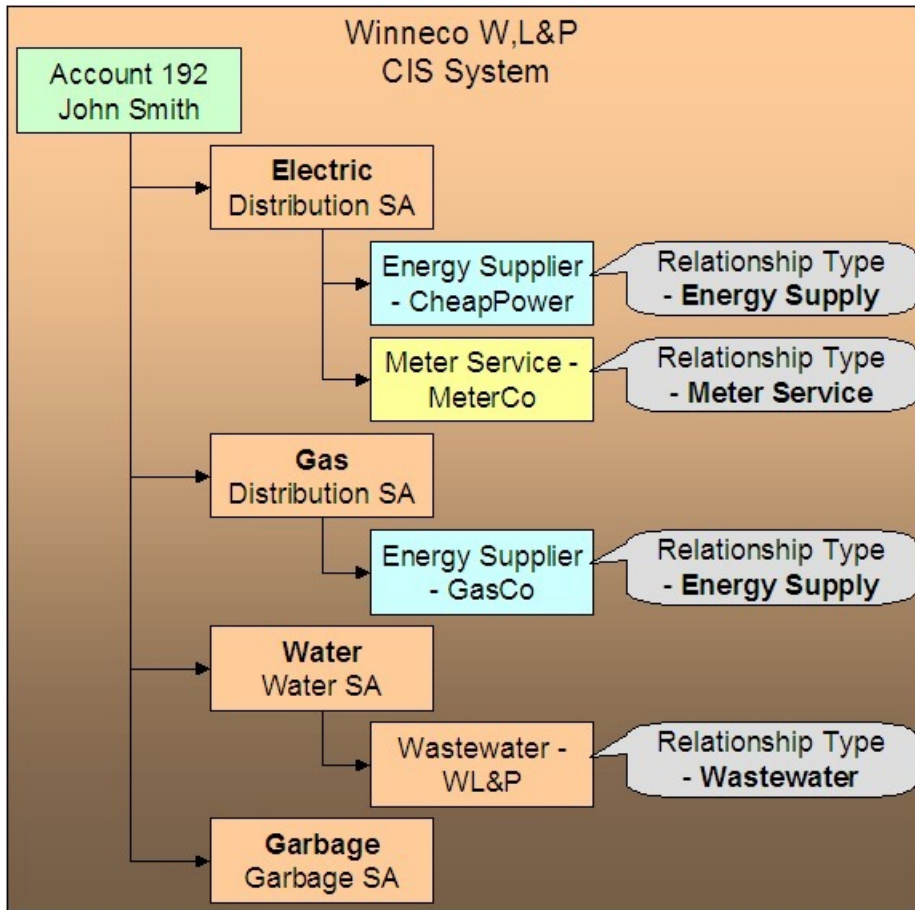
The system does nothing special to enforce or record these inter-relationships. Why? Because most service provider switches are received by notification records. Notification records indicate all the associated service providers. There is no need for the system to maintain the inter-service provider associations.

A Service Agreement Can Have Many Types Of Relationships

As described in the previous sections, a given service can be subdivided into subcategories. Each type of service can have zero, one or more subcategories. We call each subcategory a **SA Relationship Type**. The topics in this section provide information about SA relationship types.

An Example

We'll use the following example of a customer in a municipal utility's CIS system to explain SA Relationship types:



Note the following:

- Electric service has 2 SA relationship types: energy supply and meter service.
- Gas service has 1 SA relationship type: energy supply.
- Water service has 1 SA relationship type: waste water.
- Garbage service has no SA relationship types.

NOTE:

Bottom line. Service providers are related to the customer via "your" service agreement. Each service provider linked to a service agreement is defined in respect of a SA Relationship Type. This relationship is effective-dated because we care about how it changes over time.

Your relationship type is implied. The SA relationship type of "your" service agreement is implied, e.g., if you are a distribution company, "your" service agreement's implied SA relationship type is "distribution".

Valid Relationship Types and Service Providers Are Defined On SA Types

You control which services have SA relationships (and which don't) when you set up your SA types. Each SA type can have zero, one or more SA relationship types. Each relationship type, in turn, can have one or more valid service providers.

FASTPATH:

Refer to [Setting Up SA Relationships For SA Types](#) for the page used to define the service providers and SA relationship types for each SA type.

Defaulting Relationship Types And Defaulting Service Providers

Please be aware of the following:

- A SA relationship type for a given SA type can be marked as being **Required**.
- A service provider for a given SA relationship type / SA type can be marked as being the **Default**.

If, at activation time, the customer's master SA is missing a **Required** SA relationship that has a **Default** service provider, the activation process automatically creates the SA relationship type and links to it the **Default** service provider. If a master service agreement doesn't have all **Required** SA relationships, the service agreement cannot be activated. This is handy when your organization is the default service provider for a relationship type.

FASTPATH:

Refer to [Automatic Creation of Sub SAs](#) for information about how the system will automatically create sub service agreements for the defaulted SA relationship / service provider if your organization provides billing services for the service provider. Refer to [Setting Up SA Relationships For SA Types](#) for the page used to define the service providers and SA relationship types for each SA type.

Required Relationship Types and Billing

When the system attempts to create a bill segment for a service agreement whose SA type has **Required** relationship types, it checks if all such relationships are defined for the service agreement. If not, a bill segment error will be generated.

The reason this restriction exists is to handle the situation when your required relationship types change over time. For example, assume on your first day of production you only need energy suppliers defined on electric service agreements. After several months, gas deregulates. When this happens, you will need to change your control tables to indicate that your gas SA types require an energy supplier. If you don't write a default program to update your existing gas service agreements, billing will complain.

Relationship Types Do Not Impact Start / Stop

Customer service representatives (CSR's) are typically not involved with the customer's choice of service providers. Most organizations hear about a customer's service providers from the service providers or from a central body. This means that the start / stop dialog is not impacted by SA relationships. This also means that CSR's are not impacted by SA relationships (unless something goes wrong). If something goes wrong, the CSR's may need to manually correct SA relationships. Refer to [How To Set Up Service Providers On A Customer's Service Agreement](#) for more information.

Billing Relationships

When you set up a service provider, you must define your organization's billing relationship with the service provider. The following points provide examples of the billing relationships supported in the system,

- If you are an energy supply company, you may provide billing services for the distribution company. This means that your bill contains both your charges and the distribution company's charges. We refer to this as the **We Bill For Them** billing relationship.
- Alternatively, the distribution company may provide billing services for you. This means that the distribution company's bill contains your charges and their charges. We refer to this as the **They Bill For Us** billing relationship.
- Alternatively, you may both send bills to the customer. We refer to this as the **Dual Billing** relationship.
- Alternatively, if you subcategorize your services OR if your company provides one of the services that is provided by your service providers, then the system will create a separate bill segment for the subcategory of service. We refer to this as the **It's Us** billing relationship.

If you provide billing services for another service provider (i.e., you bill for them), there are two ways to determine the service provider's charges:

- You can load the service provider's rates in your system and calculate the charges for the service provider. We call this the **Rate Ready** calculation method.
- You can let the service provider calculate their own charges and interface them to you. We call this the **Bill Ready** calculation method.

If a service provider provides billing services for you (i.e., they bill for you):

- If the service provider has a suitable CIS system, they can load your rates in their system and calculate your charges for you. We call this the **Rate Ready** calculation method.
- You can calculate your charges and interface them to them. We call this the **Bill Ready** calculation method.

If you don't have a billing relationship with a service provider, you still need a service provider record to define such. Why? Because the system needs to know that it doesn't have to worry about a particular service provider in respect of billing. In addition, you may have other interactions with a service provider that have nothing to do with billing, e.g., you may send or receive consumption.

The topics in this section provide a wealth of information about the various billing relationships and the ramifications of each.

Sub Service Agreements

If you provide billing services for another service provider OR if you subcategorize your own charges, there will be a separate service agreement (SA) linked to the customer's account that holds these unique charges. We refer to this new service agreement as a **Sub SA**. We use the term "sub" because this service agreement is subservient to the "master" service agreement. By subservient we mean:

- A sub SA's start and stop dates are the same as the master SA. This statement may seem odd, but it's true – all sub SAs linked to a master service agreement have the same start and stop dates as the master.
- A sub SA's status (i.e., *pending start*, *active*, *pending stop*, *stopped*, etc.) is controlled by its master service agreement. As a rule, the master SA transitions its status first and the sub SA(s) follow.

NOTE: There are situations where the master service agreement transitions its status and the sub service agreement(s) does not. One example of this is when both the master and sub SAs are closed and a payment cancellation occurs that only affects the master SA. In this scenario, the master SA becomes reactivated and the sub SA remains closed. Another example is when a straggling billable charge posts to a sub SA; the master SA is not affected.

- Refer to [The Rate Ready Calculation Method](#) for additional examples of subservience.

NOTE:

You may find it helpful to keep in mind that sub SAs are only used for service providers with a billing relationship of It's Us and We Bill For Them. This is because these are the only relationships that have implicit billing responsibilities.

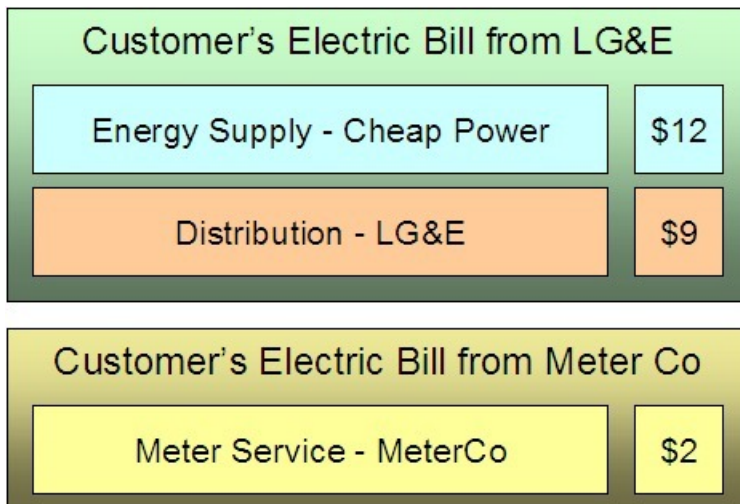
The topics in this section provide additional information about sub SAs.

Only Some Service Providers Have Sub SAs

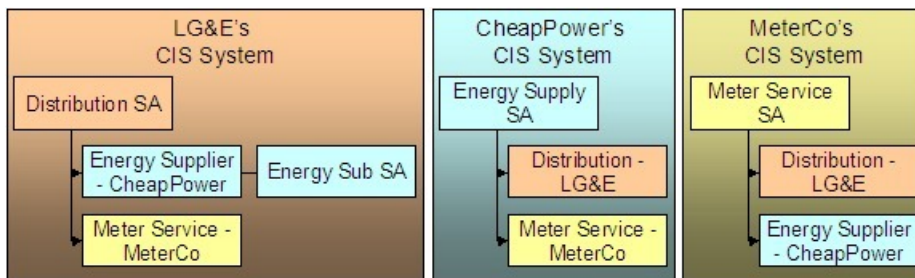
If you do not provide billing services for a service provider, there will be no sub SAs associated with the service provider's SA relationships. Let's use an example to make the point; assume:

- Service is divided into distribution, energy and meter service charges.
- You are the distribution company and you provide billing services for the energy supply company.
- The meter service company bills the customer independently.

In this situation, the customer will receive two bills: one from you (LG&E), the other from MeterCo. Notice that your bill (LG&E) contains your distribution charges AND CheapPower's energy charges:



If we were to look at the customer information in the three service providers' respective CIS systems, we'd find the following service agreements and SA relationships for the customer:



Notice the following:

- The customer has a service agreement in each supplier's CIS system.
- Because LG&E (the distribution company) bills for CheapPower, all customers who have their energy supplied by CheapPower will have a sub SA in LG&E's system. This sub SA maintains the charges (and receivable balance) associated with CheapPower's energy charges.

- Notice that neither CheapPower's nor MeterCo's CIS systems use sub SAs. This is because neither company bills for other service providers.

NOTE:

Sub SAs are needed if you subcategorize your charges. The above example shows sub SAs being used when a company provides billing service for another company. Sub SAs are also used when you subcategorize your charges - each sub SA contains the rate associated with each subcategory.

Automatic Creation of Sub SAs

The system creates sub SAs for customers choosing service providers where the billing option is It's Us or We Bill For Them. The system creates sub SAs via the following mechanisms:

- The [analyze SA relationships](#) background process (known by the batch control ID of ANLYZSAR) monitors newly activated SA relationships. If the respective service provider is It's Us or We Bill For Them, this process creates the sub SA(s) using the information defined on [SA Type SA Relationship Type - Sub SA Type](#).

NOTE:

The [analyze SA relationships](#) background process also activates and stops sub SAs. Refer to [Sub SA State Transition](#) for the details.

- A button exists on the [SA Relationships](#) page. This button, when pressed, creates sub SAs real-time. This button would only be used if the operator couldn't wait for the background process to run.

The SA type associated with the new sub SAs is defined when you set up each SA Type / SA Relationship Type. Refer to [Setting Up SA Relationships For SA Types](#) for more information.

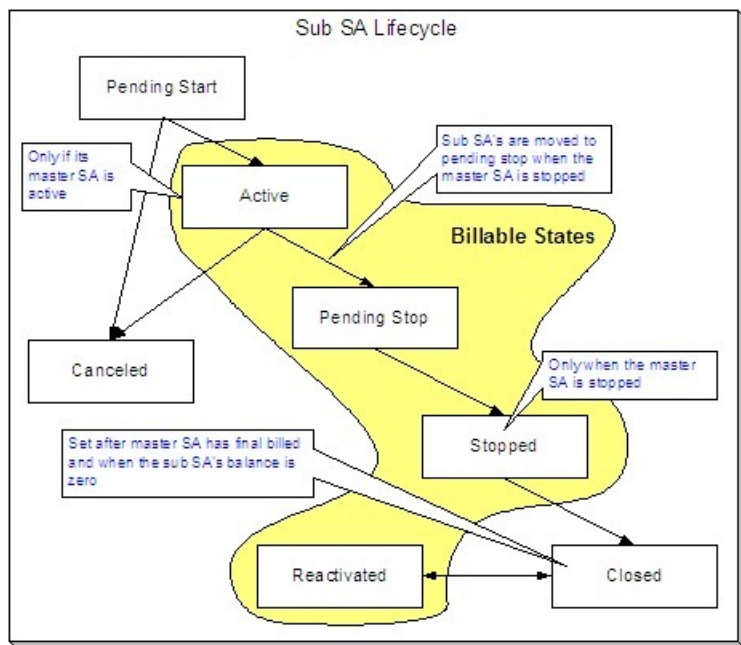
If the SA type uses start options, the start option contains additional values (e.g., rates, contract riders) that are used to populate the newly created sub SAs. Note: start options are typically not used for We Bill For Them - Bill Ready service provider because we don't need to default rates, contract riders and/or contract values on billable charge sub SAs. Refer to [Setting Up SA Relationships For SA Types](#) for more information.

NOTE:

Manually created sub SAs. In very unusual situations, an operator may create a sub SA manually. An operator would do this using the [SA Relationship](#) transaction.

Sub SA State Transition

Sub SAs follow normal SA state transition rules, but have a few additional rules about when the transitions can take place.



The following points highlight the additional state transition rules:

- A sub SA can only become Active if its master SA is Active. Most sub SAs are activated by the [analyze SA relationships](#) background process. However, a user can manually activate a sub SA using the activate button on the [SA maintenance](#) page.
- Sub SAs become Pending Stop when their master becomes Stopped . This typically occurs when the [SA activation background process](#) stops the master SA. However, this can happen real time if a user manually stops a master SA using the stop button on the [SA maintenance](#) page.
- A sub SA can only become Stopped if its master SA is Stopped . Most sub SAs are stopped by the [analyze SA relationships](#) background process. However, a user can manually activate a sub SA using the stop button on the [SA maintenance](#) page.
- A sub SA becomes Closed if its master has been final billed and it has a balance of zero. Most sub SAs are closed when their balance becomes zero after the master SA is final billed. However, a user can manually close a sub SA using the close button on the [SA maintenance](#) page. Note, if additional billable charges are interfaced after the master has been closed, the sub SA will be Reactivated.

We Bill For Them

If you provide billing services for another service provider, then you have a We Bill For Them billing relationship with the service provider. The topics in this section provide information about this type of billing relationship.

Sub SAs Are Used When We Bill For Them

As described under [Sub Service Agreements](#), sub SAs are used for We Bill For Them service providers. The sub SAs hold the service providers charges. It's important to note that the system allows more than one sub SA to be created for a given service provider.

The Rate Ready Calculation Method

If you provide billing services for a service provider (i.e., the service provider has a billing relationship of We Bill For Them), you can load the service provider's rates in your system and calculate the charges for the service provider. We call this the **Rate Ready** calculation method.

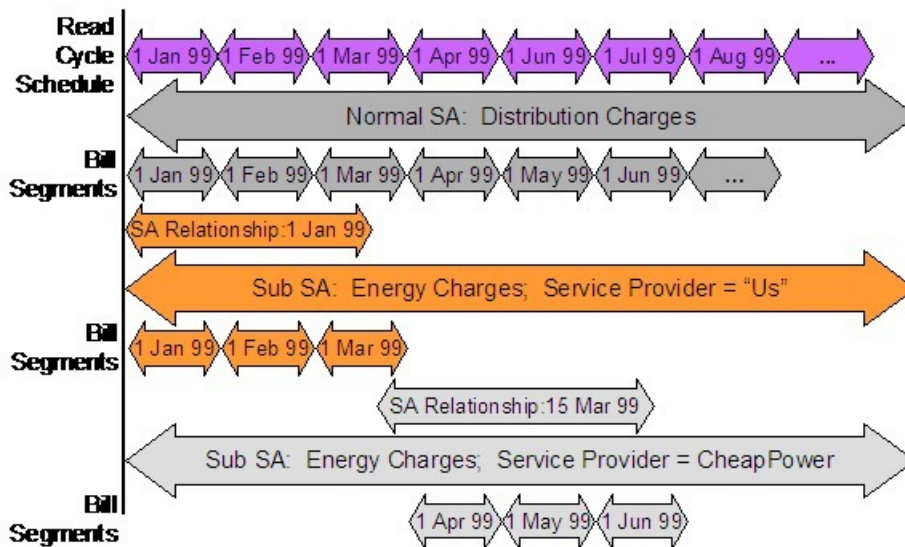
NOTE:

It's Us **can be Rate Ready**. The prior paragraph indicated that the Rate Ready billing option was used for We Bill For Them service providers. If you subcategorize your services and use rates to calculate each subcategory's charges, the service provider(s) set up for your own organization will also be subject to the Rate Ready rules described below.

If a customer uses a We Bill For Them - Rate Ready service provider, the system will create a separate bill segment for the service provider's charges. Keep in mind the following in respect of these types of bill segments:

- The consumption is copied from the "master" service agreement (note: we copy item details and read details after pre-processing calculation groups, if any exist, have been applied). This means that the bill period on the sub SA's bill segment is identical to that on the normal SA. This also means that if the "master" SA is in error, bill segments for the sub SA will not be created.
- The rate defined on the sub SA is applied against this consumption. All pre-processing calculation groups defined on the service provider's rate will be used to manipulate this consumption. If you have pre-processing calculation group on your master SA's rate to apply a line loss or convert a cubic foot to a therm, you don't have to have these pre-processing calculation group on the sub SA's rate (because we copy the consumption from the master to the sub after pre-processing calculation group, if any exist, have been applied).

The following illustration should help:



Be aware of the following:

- This example is from a distribution company's perspective.
- From 1 Jan 99 through 15 Mar 99, the distribution company distributes power AND supplies energy (i.e., the distribution company is the service provider of energy).
- On 15 Mar 99, CheapPower becomes the energy supplier. However, the next bill period ends on 30 Mar 99. This means that CheapPower will only start supplying energy on 1 Apr 99. See the following note for why the effective date of the supplier switch is not the true effective date from billing's perspective.

NOTE:

Service Provider Changes Take Effect On The Next Bill. If the service provider is changed within a customer's billing period, the system assumes that the service provider in effect at the START of the period is effective the entire billing period. This means that a change of service providers will only take effect on the bill whose start date follows the change date.

Please be aware of the following characteristics of Rate Ready bill segments:

- You cannot cancel a Rate Ready bill segment independently from the master bill segment. If you need to cancel / rebill the sub SA's bill segment, you must cancel / rebill the master SA's bill segment.
 - You cannot delete a Rate Ready bill segment independently from the master bill segment. If you need to delete the sub SA's bill segment, you must delete the master SA's bill segment.
 - Similarly, if you cancel / rebill the normal SA's bill segment, all Rate Ready bill segments will be cancelled / rebilled.
-

NOTE:

Bottom line. Creating, freezing, deleting, and canceling a "master" bill segment does the same with all Rate Ready bill segments. You cannot create, freeze, delete or cancel a Rate Ready bill segment independent of its master bill segment.

The Bill Ready Calculation Method

If a customer uses a We Bill For Them - Bill Ready service provider, the service provider calculates their own charges and interfaces them to the system. We then present their charges on our bill; we don't actually calculate anything (sometimes this is referred to as "pass through" billing).

The sub SA linked to a We Bill For Them - Bill Ready must be a billable charge SA. Why? Because billable charge SAs exist to hold bill lines until such time as a bill is created for the customer's account. Refer to [Setting Up Billable Charge Templates](#) for more information.

Please be aware of the following characteristics of Bill Ready bill segments:

- Unlike Rate Ready bill segments, Bill Ready bill segments can span different time periods than the master SA. This is because you cannot predict when a service provider will interface their billable charges to you. In fact, a given bill could contain billable charges that span different periods and these charges could have been interfaced from historic and existing service providers.
- Unlike Rate Ready bill segments, Bill Ready bill segments can be created and deleted independently from the master bill segment.
- Unlike Rate Ready bill segments, Bill Ready bill segments can be cancelled independently from the master bill segment.

[Sending Consumption And Waiting For The Charges](#)

[Uploading Consumption \(Rather Than Uploading Calculated Charges\)](#)

[Calculating Taxes On Uploaded Charges](#)

Sending Consumption And Waiting For The Charges

If your organization supports We Bill For Them - Bill Ready service providers AND you are the source of consumption used by these service provider to calculate their charges, you need to be aware of the following:

- We do NOT recommend sending raw meter reads to service providers. Rather, we recommend sending these service providers the same consumption that you use on your bill segments. Remember, the system maintains a snapshot of billed consumption on bill segments associated with service agreements that are linked to service points.

- But to implement our recommendation (of only interfacing billed consumption to service providers), we need to create a bill segment for the master SA and then wait until the service provider returns the billable charge before sending out the bill. The following points describe how this works:
 - Early in the bill cycle, the system creates a bill segment for the master SA (remember, the system maintains a snapshot of billed consumption on bill segments linked to service points). When the master bill segment is frozen, the system interfaces the snapshot consumption to all service providers associated with the master service agreement who need consumption (this is defined on the service provider). Refer to [We Can Send Billed Consumption To Any Service Provider](#) for more information about interfacing consumption to service providers.
 - The bill associated with the bill segment will not be completed. Why? Because the bill segments associated with the We Bill For Them - Bill Ready service providers will be in Error. This only happens if you use the [Billable Charge](#) bill segment creation algorithm (this algorithm is plugged in on the sub SA's SA type). On the algorithm, make sure to specify a value of Y for the parameter **Wait For The Last Day Of The Bill Cycle**. By doing this, a bill segment in the Error state will exist for the sub SA until the last night of the bill cycle's window.
 - On the last night of the bill cycle window, when the system attempts to create a bill segment for the billable charge SA associated with the We Bill For Them - Bill Ready, it will either find recently interfaced billable charges or it won't. If it finds unbilled, billable charges, a bill segment will be created for the sub SA and the billable charges will be swept onto it. If it doesn't find unbilled billable charges, the bill will be completed (i.e., sent out) without the service provider's charges.

NOTE:

Bottom line. If a customer's account uses We Bill For Them - Bill Ready service providers, the bill will not be completed (i.e., sent out) until the last night of the bill cycle (if you use the appropriate algorithm). Why? Because we wait until the last night of the bill cycle before trying to sweep on recently interfaced billable charges. If no billable charges have been interfaced from the service provider by the last night of the bill cycle, the bill will be sent out without the service provider's charges.

Batch versus Online Bill Creation. If you create a bill online, the system will NOT create an Error bill segment for the We Bill For Them - Bill Ready service provider. Why? Because if you want to create an online bill, either the service provider has interfaced their charges or they haven't. If they have, they should be swept on the bill (via the creation of a bill segment). If they haven't, it shouldn't prevent you from completing the bill.

[Top of the Page](#)

Uploading Consumption (Rather Than Uploading Calculated Charges)

We understand this is confusing, but it is quite possible to set up the system so that the We Bill For Them - Bill Ready service provider passes in CONSUMPTION rather than the calculated bill lines. They would only do this if they are not able to calculate the charges in their system and have therefore provided you with their rates. To do this, you would set up everything as described above. In addition, when you upload the billable charges, you must specify the consumption to be rated in the billable charge's service quantity (SQ) collection.

[Top of the Page](#)

Calculating Taxes On Uploaded Charges

It is possible to set up the system to calculate taxes for billable charges. You would do this if the service providers are passing through the charges and want you to calculate the taxes.

If you want taxes calculated on top of billable charges:

- Create a service quantity (SQ) on the billable charge that contains the total monetary amount that taxes will be calculated on. Note, you would not have to do this if you have a pre-processing calculation group in your rate that calculates the total monetary amount to which taxes should be applied.
- Specify a rate on the billable charge service agreement. This rate will contain calculation rules that calculate taxes. Note, the calculation rules will be simple SQ calculation rules that apply a percentage to the value of the SQ that represents the monetary amount on which taxes should be calculated.

When the system creates the bill segment for the billable charge, it will call the rate and the rate will calculate the taxes and add additional lines (actually, bill calculation headers) to the bill segment.

Pay At Bill Time vs. Pay At Pay Time

If you provide billing services for another service provider (i.e., the service provider has a billing relationship of We Bill For Them), you will owe them money because you will be receiving money from their customers for their service. You have two options in respect of when the system increases the amount you owe the service provider:

- You can tell the system to increase how much you owe the service provider when you freeze the customer's bill segment. Some people refer to this method as "purchasing the receivable from the service provider". We call this the **Pay At Bill Time** method.
- You can tell the system to increase how much you owe the service provider only when you are paid for by the customer. We call this the **Pay At Pay Time** method.

The method used for a service provider is defined on the service provider's record.

FASTPATH:

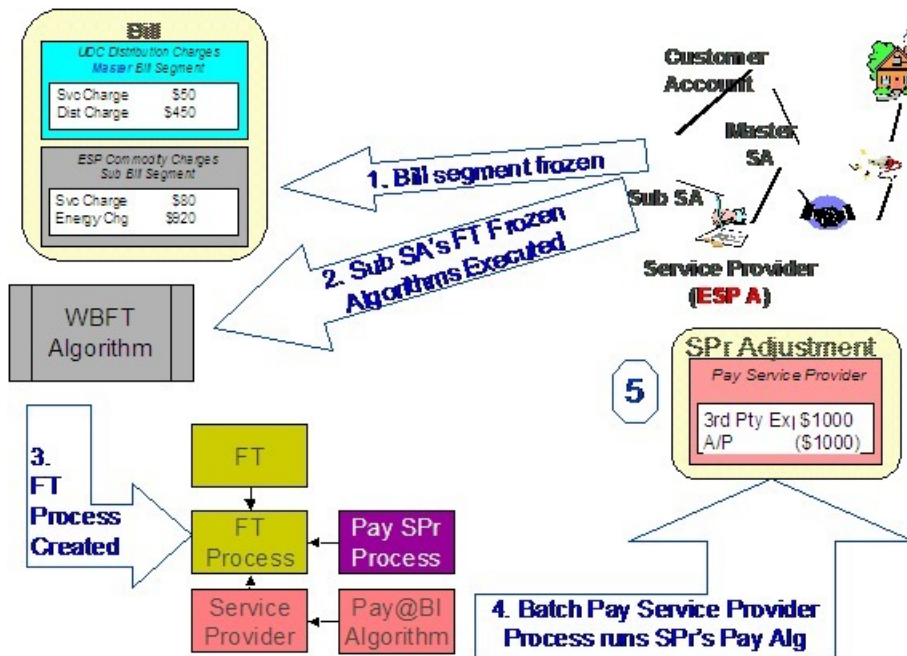
Refer to [When We Bill For Them, We Owe Them Money](#) for more information.

Paying The Service Provider - Technical Implementation

WARNING:

This section describes, technically, how we increase the amount we owe a We Bill For Them service provider. If you aren't technically inclined, skip this section.

The following illustration shows the logical steps involved with increasing how much we owe a We Bill For Them service provider.



The following points explain the steps:

- When a financial transaction (FT) is frozen, the system executes the FT Freeze algorithms defined on the SA(s) SA type.
- If you've set up the system properly (i.e., you've put the appropriate FT Freeze algorithm on the sub SA's SA type), one of these algorithms will determine if there is a WBFT service provider associated with the sub SA. If so, it will insert a row on the FT Process table.
- Rows on the FT process table are used as "triggers" for batch processes. In this case, the batch process that is triggered is the one that looks at new FT's and determines if a related "payable" adjustment should be created for the We Bill For Them service provider. This batch process uses the service provider's Payment Relationship and Pay Service Provider algorithm to determine when and how to create these "payable" adjustments.

Service Providers Have Service Agreements Too

Most service providers need service agreements as explained in the following topics.

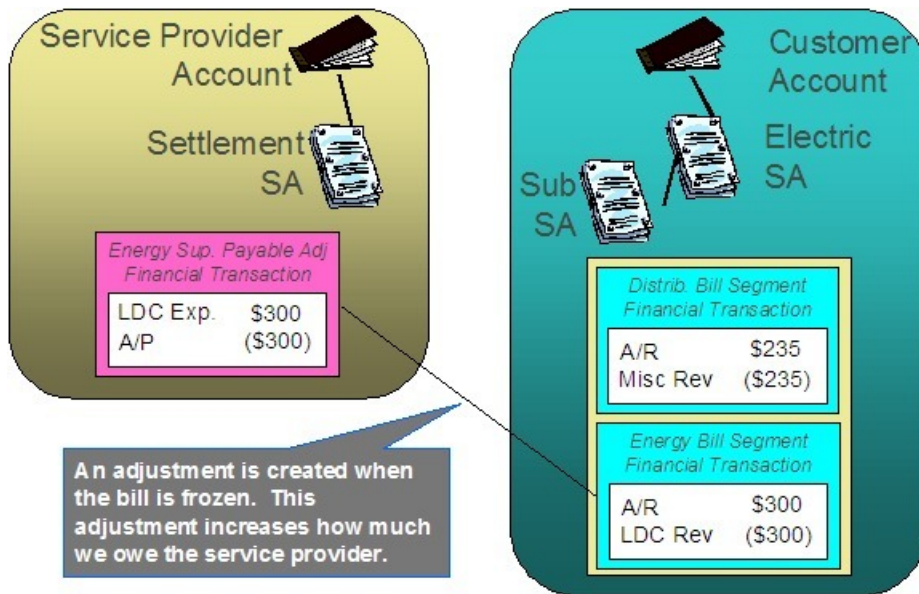
When We Bill For Them, We Owe Them Money

When you bill on behalf of a service provider (i.e., the service provider has a billing relationship of We Bill For Them), you will eventually owe them money (because the customer pays you for the service provider's service). You have two options in respect of when the system increases the amount you owe the service provider:

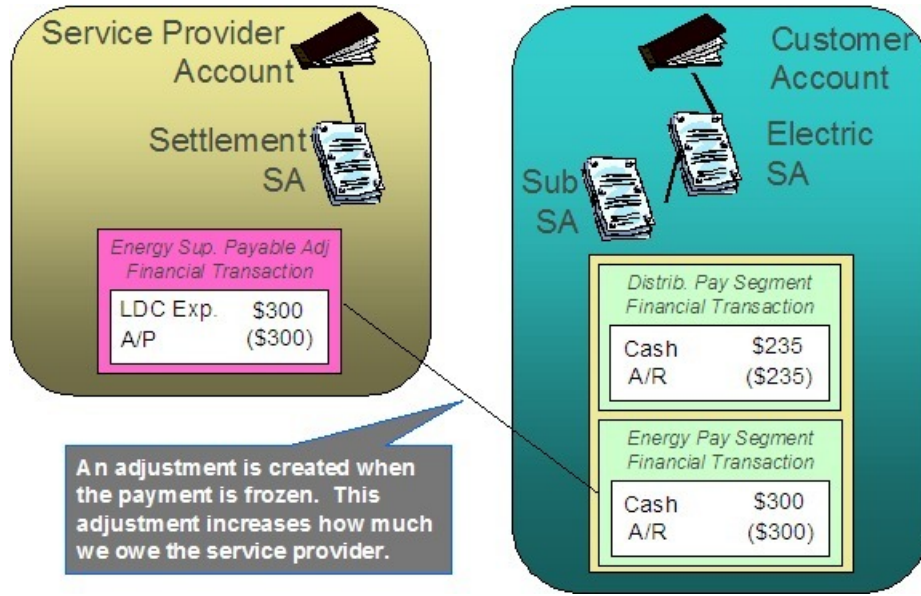
- You can tell the system to increase how much you owe the service provider when you create the customer's bill. Some people refer to this method as "purchasing the receivable from the service provider".
- You can tell the system to increase how much you owe the service provider only when you are paid for by the customer.

The system keeps track of how much you owe a service provider on a service agreement linked to the service provider's account. The system creates adjustments against this service agreement to increase how much you owe them.

If you "purchase the receivable" (i.e., you owe them when you bill the customer), an adjustment is created when the customer is billed. If you owe them only when you are paid by the customer, an adjustment is created when the customer pays. The following example illustrates an adjustment being created when the bill is frozen (illustrating the "purchase the receivable" scenario):



If you only pay the service provider when you are paid, the example would look as follows:



Adjustments and We Bill For Them Service Providers

FASTPATH:

It's important that you are comfortable with the information described under [When We Bill For Them, We Owe Them Money](#) before reading this section.

Adjustments associated with We Bill For Them sub service agreements are tricky. The following points describe how the system "pays" the related service provider when adjustments are issued against the customer's sub service agreement:

- For Pay At Bill Time service providers, most adjustments are treated just like bill segments, i.e., when the adjustment's FT is frozen, a payable adjustment is created for the respective service provider. The reason "most" is underlined in the previous sentence is because A/P adjustments (i.e., adjustments used to interface check requests to your A/P system) are excluded. Why? Because A/P adjustments are used to refund overpayments to the customer. Overpayments are purely between the customer and your company (you never transferred the overpayment to the service provider because it's associated with a Pay At Bill Time service provider).
- For Pay At Pay Time service providers, A/P adjustments are treated just like payment segments, i.e., when the adjustment's FT is frozen, a payable adjustment is created for the respective service provider. All other types of adjustments are ignored. Why? Because A/P adjustments are used to refund overpayments to the customer. Think of it like this - when the customer originally overpaid, you transferred this overpayment to the Pay At Pay Time service provider; therefore, when you refund the overpayment, you get to take the money back from the service provider.

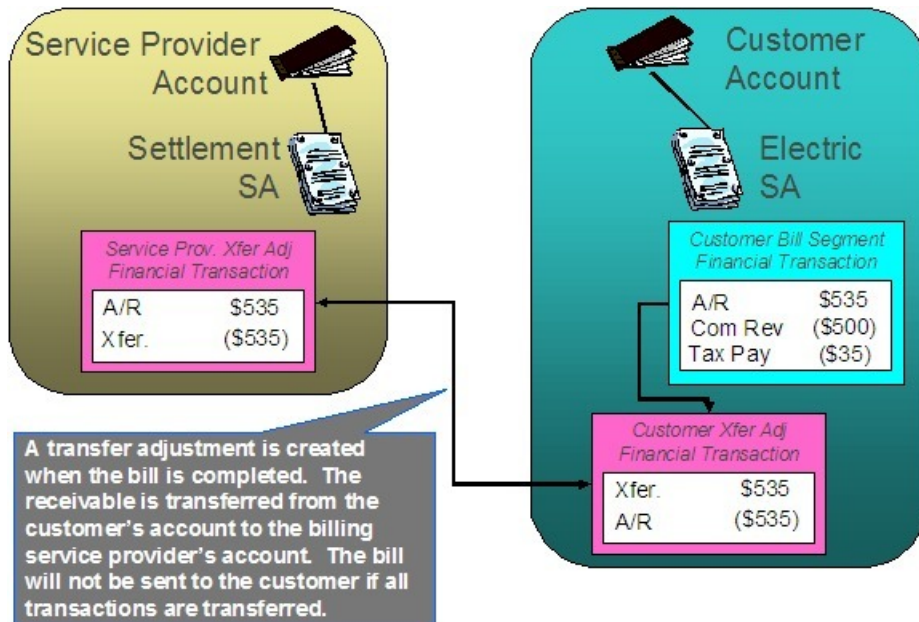
When They Bill For Us, They Owe Us Money

When a service provider bills on behalf of your organization, they will eventually owe you money (because the customer pays them for your service).

The system keeps track of how much a service provider owes you on a service agreement linked to the service provider's account. The question is, How does the system determine how much you are owed when you don't produce a bill? Well, you do produce a bill, it just doesn't get sent to the customer. We understand this is a little confusing, but think about it like this:

- The service provider is presenting your charges on their bill.
- You still have to calculate how much the customer owes your organization otherwise you'll never know how much you are owed by the service provider.

As illustrated below, when a bill is completed, the system determines if there are bill segments and/or adjustments associated with service agreements with a service provider who "bills for us". If it finds these, it transfers the receivable from the customer's service agreement to the service provider's service agreement. If all financial transactions have been transferred to the service provider, no bill is produced for the customer.



NOTE:

Bottom line. We always generate a bill for "us", even though we don't send it to the customer.

They Bill For Us

If a service provider provides billing services for you, then you have a They Bill For Us billing relationship with the service provider. The topics in this section provide information about this type of billing relationship.

The Customer Still Needs A Service Agreement

If a service provider bills for you, you still need a service agreement for the customer. Why? Because:

- As explained under [Service Providers Are Linked To Service Agreements](#), service providers are defined in respect of a customer's service agreement (therefore the customer must have a service agreement).
- As explained under [When They Bill For Us, They Owe Us Money](#), you still have to calculate bills for the customer.

NOTE:

A customer's bill history still exists. Be aware that even when a service provider bills for us, you will still be able to see the customer's billing history. It's just that the customer won't owe you anything because the receivable balance will be transferred to the service provider's account.

They Bill For Us - Bill Ready

At bill completion time, the system determines if there are bill segments and/or adjustments associated with service agreements with a service provider who "bills for us". If it finds these,

- It transfers the receivable from the customer's service agreement to the service provider's service agreement. If all financial transactions have been transferred to the service provider, no bill is produced for the customer.
- Note that payments and A/P adjustments are not transferred. Why? Because payments and A/P adjustments are purely between the customer and your company.
- Each bill segment and adjustment is marked to be interfaced to the service provider (via a separate background process).

They Bill For Us - Rate Ready

The They Bill For Us - Rate Ready option is not a recommended option. Why? Because you really have to compute how much the customer owes as explained above. If you go to the trouble of figuring out how much the customer owes, then it makes sense to interface this to the billing service provider.

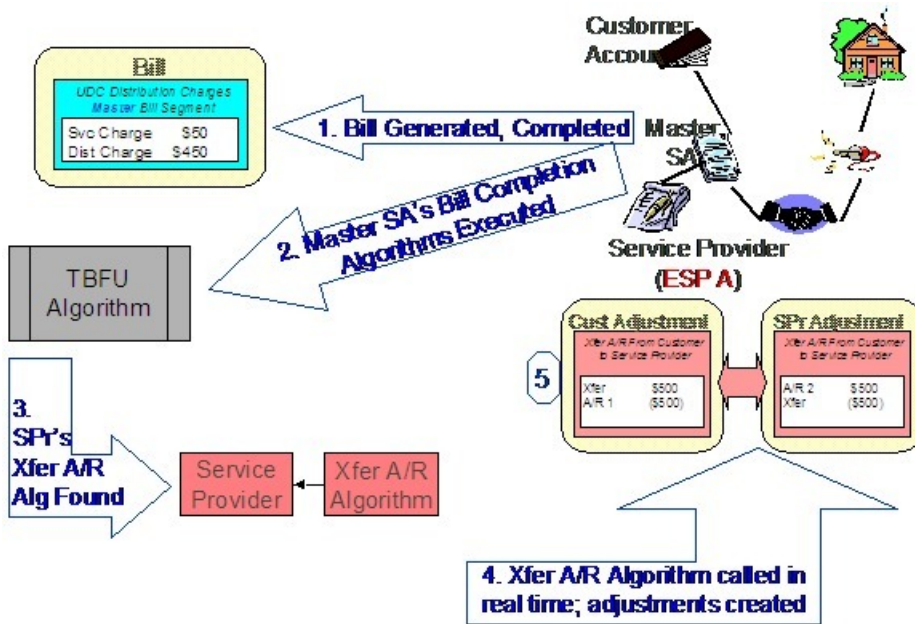
The only difference between this option and They Bill For Us - Bill Ready is that the system will not interface the bill segments and adjustments to the billing service provider.

A/R Transfer - Technical Implementation

WARNING:

This section describes, technically, how a customer's A/R balance is transferred to a They Bill For Us service provider. If you aren't technically inclined, skip this section.

The following illustration shows the logical steps involved with the transference of a customer's A/R balance to a They Bill For Us service provider.



The following points explain the steps:

- When a bill is completed, the system executes the bill completion algorithms defined on the bill's master SA(s) SA types.
- If you've set up the system properly (i.e., you've put the appropriate Bill Completion algorithm on the master SA's SA type), one of these algorithms will determine if there is a They Bill For Us service provider associated with each master SA on the bill. If so, it will execute the Transfer A/R algorithm defined on the service provider's record. This algorithm causes a transfer adjustment to be created (transferring the financial transaction's affect on the customer's balance from the customer to the service provider).

NOTE:

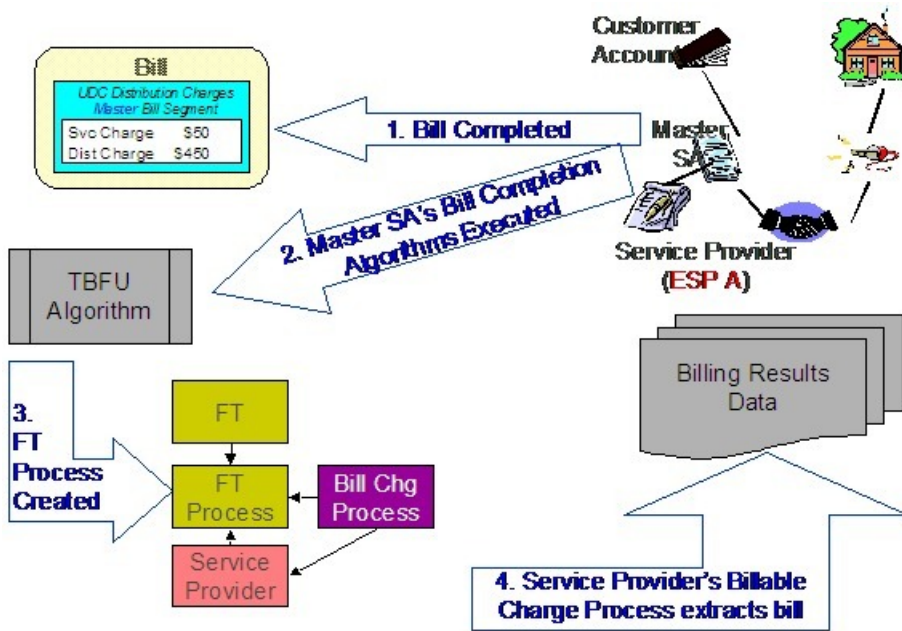
If there are multiple master SAs on a bill, the financial details associated with each respective master SA could be transferred to different service providers (e.g., one service provider could receive the financial details for gas and another for electricity). Refer to [Different Service Providers Can Bill Different Services](#) for more information.

Routing Billable Charges To Service Providers - Technical Implementation

WARNING:

This section describes, technically, how we send billable charges to service providers. If you aren't technically inclined, skip this section.

The following illustration shows the logical steps involved with sending billable charges to service providers.



The following points explain the steps:

- When a bill is completed, the system executes the bill completion algorithms defined on the bill's master SA(s) SA types.
- If you've set up the system properly (i.e., you've put the appropriate Bill Completion algorithm on the master SA's SA type), one of these algorithms will determine if there is a They Bill For Us service provider associated with each master SA on the bill. If so, it will insert a row on the FT Process table.
- Rows on the FT process table are used as "triggers" for batch processes. In this case, the batch process that is triggered is the one that downloads billable charges to the service provider. The ID of the batch process that is referenced on the trigger comes from the Service Provider's Billable Charge Download Process.

NOTE:

If there are multiple master SAs on a bill, the financial details associated with each respective master SA could be routed to different service providers (e.g., one service provider could receive the financial details for gas and another for electricity). Refer to [Different Service Providers Can Bill Different Services](#) for more information.

Bill Routings Are Changed

If all of an account's "master" SAs have a SA relationship with a service provider who bills for us (i.e., the service provider's billing relationship is They Bill For Us), then we have nothing to send to the customer. The system still creates bill routings, but with a couple of differences:

- The Batch Process Id and Run Number are reset.
- The Customer's Name is prefixed with the text from a bill message code 6, 10103. This message code's text is *** Bill not sent.

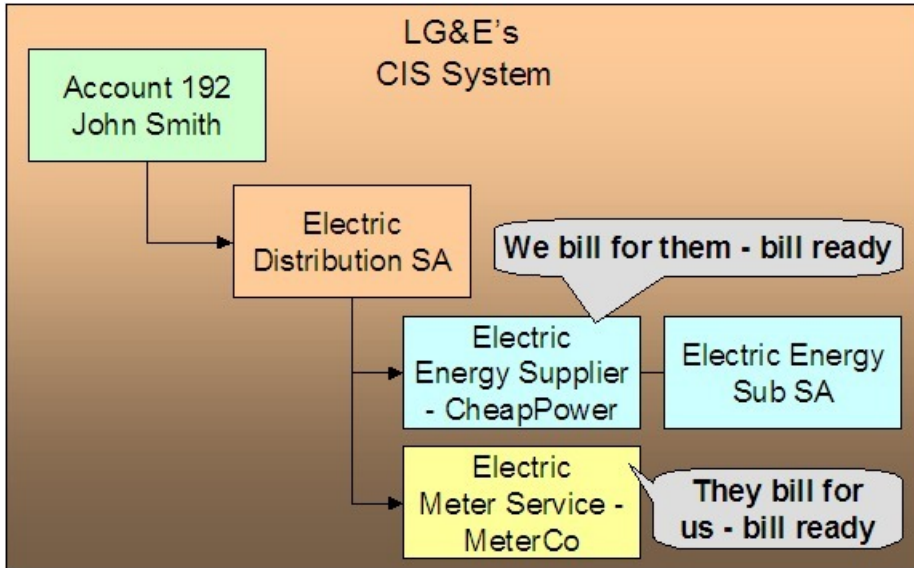
This way, the operators can easily see that that the bill was not routed and why.

FASTPATH:

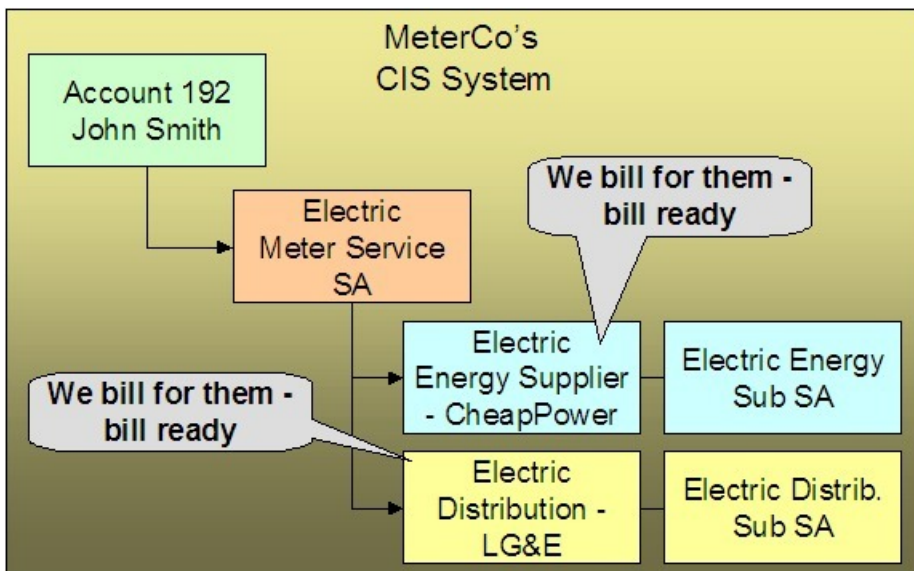
Refer to [Different Service Providers Can Bill Different Services](#) for information about how the various services under an account could be billed by different service providers.

Combinations Of Service Provider Billing Methods

Consider the following situation:



In the above example, we are billing for the energy service provider and the meter service provider is billing for us. This means that we will calculate the charges for ourselves, interface charges from CheapPower, and then interface our charges and CheapPower's charges to MeterCo. MeterCo will then produce a bill for the customer that contains our distribution charges, CheapPower's energy charges, and MeterCo's service charges. To help solidify this point, let's look at how this customer would look in MeterCo's CIS system.



There are some restrictions in respect of permissible combinations of service providers that may supply service to a customer as described in the following points:

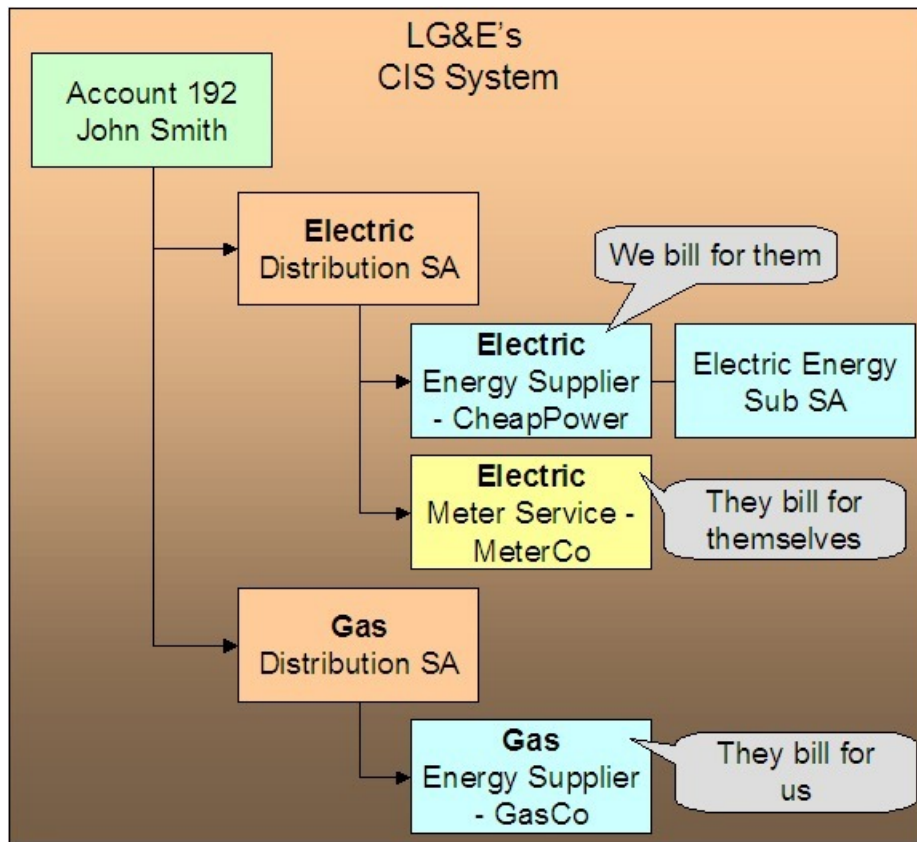
- If the system encounters a customer with a We Bill For Them (WBFT) - Bill Ready service provider and another service provider that is They Bill For Us (TBFU) - Rate Ready, a billing error will be produced. Why? Because TBFU - Rate Ready means they have everything they need to calculate our bills and therefore we do not interface bill lines to them.

If we don't interface our bill lines to them, then we can't interface the charges that were interfaced from the WBFT - Bill Ready service provider. You may wonder why we don't prohibit WBFT - Rate Ready and TBFU - Rate Ready, because it's conceivable for the TBFU service provider to have our rate and the WBFT service provider's rate.

- If the system encounters a WBFT - Pay At Pay Time service provider and another service provider that is TBFU, a billing error will be produced. Why? Because when the system detects a TBFU service provider, it transfers the receivable from the customer to the service provider (and therefore the customer's account will never be paid).

Different Service Providers Can Bill Different Services

Be aware that the system determines billing relationships at the service agreement level, NOT at the account level. To make the point, check out the following customer in a distribution company's system:



Be aware of the following in respect of the above illustration:

- The distribution company (LG&E) distributes both electricity and gas.
- The customer has a choice of energy service providers for both gas and electricity.
- This customer - John Smith - purchases his electricity from CheapPower and his gas from GasCo.
- LG&E provides billing services for CheapPower.
- GasCo provides billing service for LG&E.

In this situation, LG&E will send bills to the customer that contain both electric distribution and energy charges (but no gas distribution charges). GasCo will also send bills to the customer; these will contain both LG&E's gas distribution charges as well as their own energy charges.

If You Deal With TBFU Service Providers, You Cannot Reopen Bills

FASTPATH:

Refer to [Bill Lifecycle](#) for information about reopening previously completed bills.

If your organization deals with They Bill For Us (TBFU) service providers, a great deal happens when a bill is completed (e.g., the receivable is transferred from the customer to the service provider, we may mark the bill segments and adjustments for routing to the service provider, etc.). These things cannot be undone and therefore the system will not let you reopen bills when these things have occurred.

NOTE:

Technical rule. The specific rule that prevents the reopening of bills is as follows: if a bill contains a service agreement whose SA type has one or more bill completion algorithms, the system will not allow the bill to be reopened. Refer to [SA Type - Algorithms](#) for more information about bill completion algorithms.

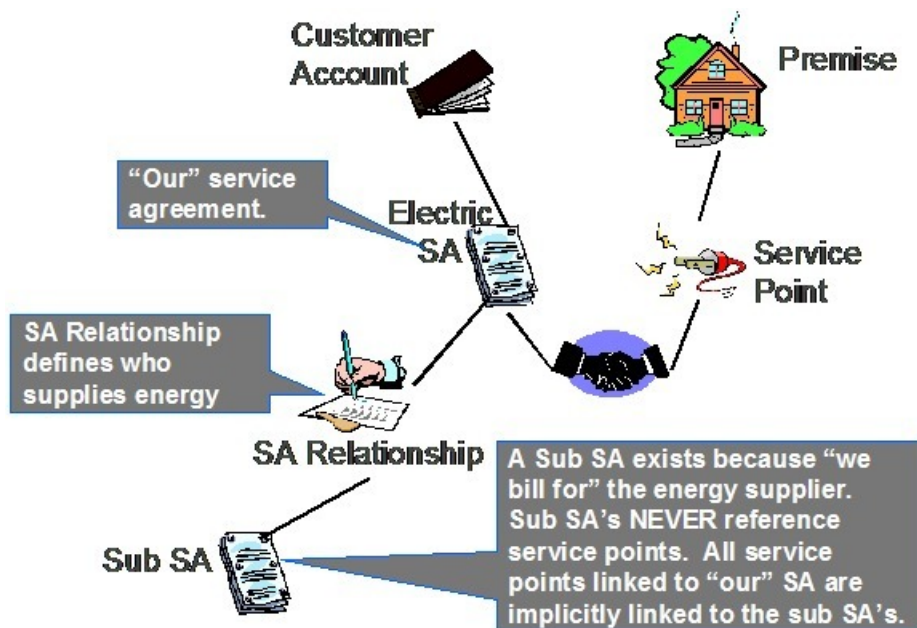
Consumption Relationships

When you set up a service provider, you must define your organization's consumption relationship with the service provider - a service provider may send the customers' consumption to you, you may send consumption to them, or you may have no consumption relationship with a given service provider.

The topics in this section provide a wealth of information about the various consumption relationships and the ramifications of each.

Only The Master SA Is Linked To Service Points

The following diagram makes the point that service points cannot be linked to sub SAs (i.e., service agreements that exist to hold charges associated with We Bill For Them service providers). This is because all service points associated with our "master" SA are implicitly linked to all sub SAs.



It's important to understand why the system does not allow sub SAs to reference service points:

- The consumption associated with all service providers should be the same, otherwise the customer will receive inconsistent bills from different service providers.
- The easiest way to ensure consumption is the same for all service providers is to make sure that they all have the same service points (which they must if sub SAs "inherit" their service points from their master).

We Can Send Billed Consumption To Any Service Provider

You can send consumption to any service provider. It doesn't matter what their billing relationship is. Information on the service provider object tells the system if AND how to send consumption to a service provider.

Rather than send raw reads to service providers, we download consumption that has been calculated and snapshot onto the "master" bill segment. We send this billed consumption because:

- It is clean and validated
- Register indexes have been subtracted
- Multiple registers have been summed

NOTE:

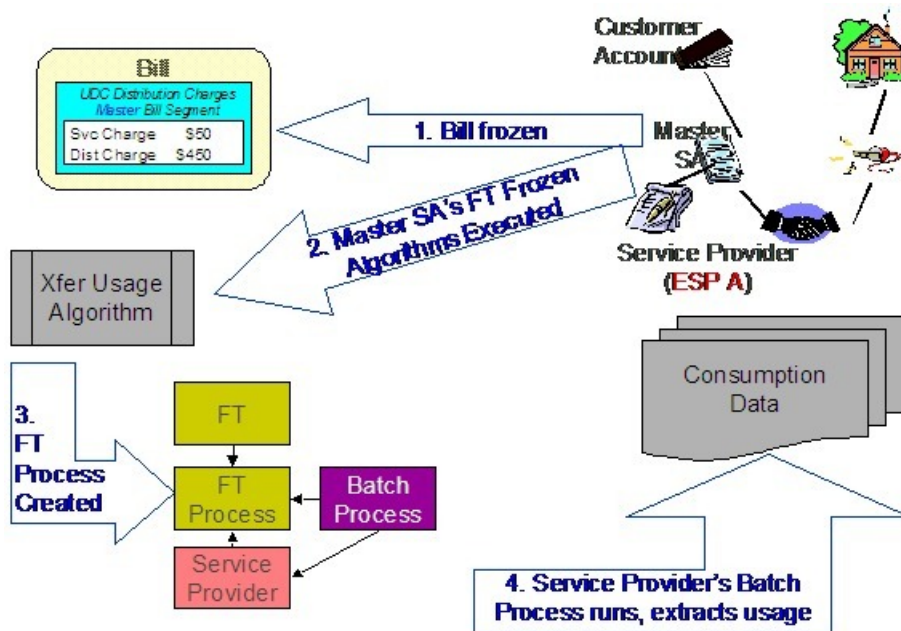
We only send consumption after a bill segment is Frozen. If your organization supports We Bill For Them - Bill Ready service providers AND you are the source of consumption used by these service providers to calculate their charges, please refer to [The Bill Ready Calculation Method](#) for an explanation of how the system waits a given amount of time for the recipient of the consumption to interface their charges back to the system before sending the bill to the customer. In other words, the bill sent to the customer should contain the bill segment that triggered the consumption download as well as bill segments containing uploaded billable charges.

Routing Consumption To Service Providers - Technical Implementation

WARNING:

This section describes, technically, how we send consumption to service providers. If you aren't technically inclined, skip this section.

The following illustration shows the logical steps involved with sending consumption to service providers.



The following points explain the steps:

- When a financial transaction (FT) is frozen, the system executes the FT Freeze algorithms defined on the SA(s) SA type.
- If you've set up the system properly (i.e., you've put the appropriate FT Freeze algorithm on the master SA's SA type), one of these algorithms will determine if there are service providers associated with the master SA who need consumption. If so, it will insert a row on the FT Process table.
- Rows on the FT process table are used as "triggers" for batch processes. In this case, the batch process that is triggered is the one that downloads billable consumption to the service provider. If multiple service providers need consumption, multiple rows will be inserted. The ID of the batch process that is referenced on the trigger comes from the Service Provider's Consumption Download Process.

We Can Receive Consumption From Service Providers

We can receive consumption from any source. We use the standard meter read upload to interface consumption from service providers. Because we use the standard meter read upload, each customer must have:

- A premise
- A service point
- A meter with registers sufficient to hold the interfaced consumption

NOTE:

You don't have to perfectly model the service points and meters. If you are not the distribution company, you may be worried about how to keep meter and service point information up-to-date. Be aware that you don't have to model the SP's and meters perfectly. Why? Because you just need to set up enough information so that consumption can be uploaded accurately. Let's use an example of a customer with multiple meters and service points at a premise. Rather

than model this perfectly, you could set up a single SP and link to it a single meter. All that matters is that the meter has the appropriate registers to hold the interfaced consumption.

When a customer's consumption (i.e., meter reads) is uploaded from a service provider, the service provider must be defined as the "upload source" on the meter read upload staging records. The system uses this information to validate that the service provider is linked to the customer on the effective date of the read AND that they are defined as Sending Consumption To Us. Refer to [Uploading Meter Reads](#) for more information.

MDMAs And Service Cycles

WARNING:

This section uses terminology and concepts described in [The Cyclical Meter Read Process](#).

A Meter Data Management Agency (MDMA) is a service provider who reads meters. In some locales, a meter can be read by a variety of MDMAs, in other locales there is no MDMA as the meter is read by the distribution company. If your organization has MDMAs, then you need to be aware of the following:

- MDMAs must be linked to service agreements as a service provider. Like all service providers, they can change over time.
- When an MDMA is reading a service agreement's meters, the MDMA may override the customer's service cycle (and schedule) with their own cycle. If they do this, the customer's service cycle in the MDMA's system is defined on the service agreement's SA relationship information for the MDMA. Note: whether or not a service provider can override a service cycle is controlled by a switch on the service provider's record. Refer to [Service Provider - Main](#) for more information.
- A service provider's service cycle schedules are maintained using the system's normal service cycle schedule. The ID of the service provider associated with each schedule is defined on the service cycle; in other words, if a service provider reads meters and they can override the customer's service cycle, the service provider's service cycles must be defined in the system. Refer to [Designing Service Cycles, Routes, And Schedules](#) for more information.
- At billing time, the system determines if a service agreement is covered by an MDMA. If so, it uses the service cycle defined on the service agreement's MDMA SA relationship record.
- If your organization ever reads the customers' meters, your regular read cycle should be maintained on your service points. You can think of the service cycle that is defined on a service agreement's MDMA SA relationship record as an override of your service points normal service cycle.
- If the meter read download process detects that a service point is linked to a service agreement with an active MDMA, it still creates a meter read download staging record; however, it marks it as Do Not Need To Read. This means that when a service point is no longer read by an MDMA, the meter read will be downloaded normally.
- Refer to [We Can Receive Consumption From Service Providers](#) for a description of how a service provider interfaces consumption into the system.

Deposits Issues

WARNING:

This section uses terminology and concepts described in [The Big Picture Of Deposits](#).

Deposits should be held using normal deposit service agreements (SAs). You should NOT use the [Sub Service Agreements](#) (sub SA) functionality to hold or bill for deposits because deposit service agreements do not have the same state transition as do master SAs (e.g., you can activate or stop a deposit independent from its master).

However sub SAs can be covered by a deposit. If so, their SA type must reference a deposit class. To make the point, let's examine a few scenarios:

- Assume you have a sub SA for your own charges (this can happen when we use sub SAs to unbundle charges from the Master SA). In this case, it is likely that the sub SA and master SA will be in the same deposit class. This means that a single deposit SA would cover both the master and the sub SA.
- Assume They Bill For Us (Bill or Rate Ready). In this situation, we still have a master SA for our charges and we transfer the charges to the service provider who does the billing. In this case, it is likely that we would be holding the deposit for the service provider, not on the end-use customer. If we are in a situation where 1) we cannot hold a deposit against the service provider, and/or 2) we are not assured of the service provider paying us when the customer doesn't pay them, then we might want to put the master SA in a deposit class and hold a deposit against the customer's account using a normal deposit SA. We would not expect the service provider to bill the customer for the deposit, so we don't need a sub SA. We bill the customer directly for the deposit using our normal deposit SA.
- Assume We Bill For Them (Bill or Rate Ready). In this situation, we could hold a normal deposit SA for the customer's master SA. For sub SAs, we have two scenarios:
 - **We pay at billing time.** Since we purchase the receivable, we would want to increase our normal deposit to cover the Sub SA. To do this, the sub SA's SA type's deposit class should be the same as our master SA's deposit class.
 - **We pay at payment time.** It seems unlikely that we would want to hold a deposit on behalf of a service provider when we don't purchase the receivable. However, it is possible to do so by putting the sub SA into its own deposit class. If you did this, the system will require a separate deposit SA for the service provider's deposit. The system would calculate and refund such deposits using the algorithms defined on the new deposit SA's SA type's deposit class. It's important to be aware that the deposit is not held with respect to the specific service provider. Rather, it is just held in the system as separate deposit that could be used for any service agreement that belongs to its deposit class.

Credit and Collection Issues

WARNING:

This section uses terminology and concepts described in [The Big Picture Of Credit & Collections \(C&C\)](#).

C&C is only tricky if you deal with We Bill For Them service providers. We'll run through the service provider billing relationships to explain why:

- If we have a Dual relationship with a service provider, we don't have their debt, so we only have a responsibility to tell them when we cut a customer (via a Notification). We don't have to worry about collecting their debt.
- If we have a They Bill For Us relationship with a service provider, there is no debt on the customer's SA because it gets transferred to the service provider (and the service provider's SA will fall into arrears if they don't pay us).
- If we have a We Bill For Them or It's Us relationship with a service provider, the customer's debt associated with the service provider's service is maintained on a sub SA (i.e., it is segregated from our debt). This segregation of debt is both a powerful feature and a cause of administrative difficulties. The topics in this section provide more information about this issues.

Debt Class Recommendations

A service agreement's debt class is an important element in determining how a customer's debt is collected. In general, we recommend the following:

- If the service provider has a billing relationship of It's Us, we recommend the sub SAs belong to the same debt class as the "master". Why? Because both SAs' overdue debt should probably be grouped together under a single collection process.
- If you buy the receivable from the service provider (i.e., the service provider has a payment relationship of Pay at Billing Time), we recommend the sub SAs belong to the same debt class as the "master". Why? Because both SAs' overdue debt should probably be grouped together under a single collection process.
- If you don't buy the receivable from the service provider (i.e., the service provider has a payment relationship of Pay at Pay Time), you may want to use a different debt class on the sub SA. Why? Because you may collect the service provider's debt differently.

NOTE:

Bottom line. If both the "master" and the sub SAs fall into arrears, you will have 1 or 2 collection processes, it all depends on the debt class assigned to each SA type.

FASTPATH:

Refer to [Automating Your C&C Activities](#) for information describing how debt class plays a part in this processing.

Severing Service

Sub SAs and severance is tricky. Why?

- Because it's possible for the master SA to be in arrears when the sub SA isn't (for all the standard reasons - directed payments, cancel / rebills, etc.).
- Because it's possible for the sub SA to be on one collection process and the master to be on another (due to different debt classes or different time lines).

Both of these situations could result in severance starting for only one of the service agreements in the master / sub relationship. However, **YOU CAN'T CUT SERVICE FOR ONE WITHOUT CUTTING THE OTHER** because there is only one service point.

Before we describe how to deal with this conundrum, we'd like to remind you that the system starts a unique severance process for each SA (sub or normal) to be severed. It only creates a severance process for those service agreements linked to a collection process when the collection process' Start Severance event is activated. The type of severance process that is created is controlled by each service agreement's SA Type's severance criteria. Please keep in mind the following when designing these severance processes:

- **Only The "Master" Service Agreement Is Linked To Service Points.** This means only master SAs should have a "cut for non payment" severance event. Note: typically, such a severance process will expire the "master SA" several days after the cut event if funds are NOT received.
- As described under [Sub SA State Transition](#), a sub SA becomes Pending Stop (and eventually Stopped) when its "master" is stopped. This means the sub SAs will be finalized when the master is finalized.
- If you start severance on a sub SA when the master isn't being severed, you have a problem because you can't cut the sub SA independent from the master SA.

We'll use an example to illustrate how you should design your severance processes to deal with the above challenge. Assume you have a master and a sub SA where both are being managed under the same collection process. Also assume that the Start Severance event kicks off on 18-Dec-1999. In this situation, we'd recommend the following severance processes to be kicked off.

Master SA Severance Process		
Trigger Dt.	Event Type	Status
18-Dec-99	Set door hanger	Pending
Pending	Cut for non pay	Pending
Pending	Expire SA	Pending

Sub SA Severance Process		
Trigger Dt.	Event Type	Status
3-Jan-00	Create To Do Entry	Pending

Notice that the sub SA's severance process contains a single event that generates a To Do Entry on a date in the future of the Expire SA event on the Master SA. This entry should be something like "sub being severed independent of its master". This event will only be triggered if the master SA is paid off and the sub SA isn't. Why? Because if the master SA's Expire SA event is executed, the Sub SA will be Stopped and stopping a SA cancels outstanding severance processes. If the sub gets paid, the system will cancel the sub's severance process.

Let's change the example and assume that the master starts severance and the sub doesn't. In this situation, the master SA will eventually hit the Expire SA event and the sub SA will also stop. There's no alternative.

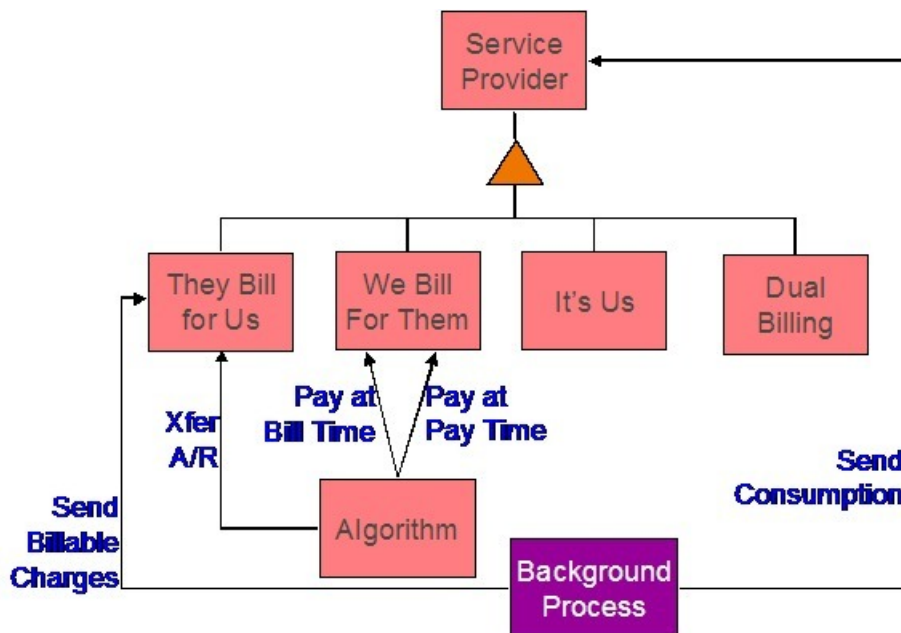
And let's change the example again and assume the sub starts severance and the master doesn't. In this situation, the To Do Entry will only be created X days after the start of severance. If you can't stand this date being X days in the future of the creation of the severance process, create an "Severance Criteria Algorithm" that checks if the master is not being severed or collected and generates a different severance process (with a different start date). Refer to [Designing Your Severance Procedures](#) for more information about Severance Criteria Algorithms.

An Object-Oriented Perspective Of Service Providers

WARNING:

Skip this section if you are not technically inclined.

The following object diagram illustrates in a concise format the various types of service providers and the plug in algorithms and processes available for each.



How Do You Communicate With Service Providers?

You communicate with service providers (and they communicate with you) using the following mechanisms:

- You can use traditional interface techniques. For example, if you send consumption to service providers every month (so they can compute their charges), you could use the Consumption Download interface.
- You can use "notifications" to communicate with service providers. Notifications are electronic transactions that service providers exchange to communicate information about a customer. For example, you could use a notification to send a message to an energy supplier when a customer stops service. Note: the term "Direct Access Service Request" (DASR) is synonymous with Notification.

The difference between notifications and traditional interfaces is subtle. Think of notifications as a generic interface that can be used to communicate many different things (e.g., you can use notifications to advise stops, meter exchanges, supplier switches). Traditional interfaces communicate only one thing (and you need one traditional interface for each "thing"). For example, one interface is devoted to downloading consumption, another is responsible for uploading pass through charges, etc.

FASTPATH:

Refer to [The Big Picture Of Notification Processing](#) for more information about notification.

Off Cycle Switch

If the customer care and billing application integrates with meter data management (MDM) system, it may be set up to generate sub-bill segments for each and every service provider having **We Bill for Them - Rate Ready** or **It's Us** billing relationship, that is effective in the billing period. This requires that the meter data management system can calculate the consumption for each effective SA relationship period.

If the switch is from or to a service provider having **They Bill for Us** billing relationship, separate bills have to be generated manually for each relationship effective in the billing period. This restriction is due to requirement for CCB application to generate a bill for a **They Bill for Us** relationship, so that the customer's charges are calculated and eventually transferred to the service provider. The generated bill for **They Bill for Us** billing relationship is not sent to the customer.

Billing Master Configuration

To enable Off Cycle Switch, navigate to **Admin > General > Master Configuration** and add a new entry for Billing Master Configuration.

Specify the values for **Off Cycle Switch Option**, **Implementation Date** and **Off Cycle Switch Option for Old Bill**. Refer to the embedded help on billing master configuration portal for details.

Rate Schedule

Rate schedules assigned to the sub-SA of service provider, having **We Bill for Them - Rate Ready** or **It's Us** billing relationship, must be configured with an SQ rule (for component-based engine) or a pre-processing calculation rule (for rule-based engine) that will obtain the sub-usage's consumption and load it into the bill segment SQ collection during rate application.

For more information, refer to the base package algorithm types provided for this purpose: [US](#) (component based engine) and [C1-ACRSUBUSG](#) (rule based engine).

Sub-SA Type

When implementing this feature, sub-SA types of service provider having **We Bill for Them - Rate Ready** or **It's Us** billing relationship must be configured as non-bill determinant.

Usage Request

Usage request is sent to the Oracle Utilities Meter Data Management (MDM) system, to obtain the calculated consumption for the billing period - this is used to calculate master bill segment's charges.

In addition, sub-usage request is sent to the MDM system, for each relationship that is effective during the billing period. Calculated consumption on these sub-usage requests is used to calculate sub-bill segment charges.

Sending Consumption to Service Providers

Calculated consumption on these sub-usage requests also serves as the source for consumption download process if consumption is required to be sent to the service provider. A base package algorithm type - [C1-STGCNSMP](#), stages the download by creating an entry in the general process table, capturing the sub-usage id, batch control for consumption download and the batch number.

NOTE: It is implementation responsibility to build the consumption download batch process that will extract the consumption for sub-usage request as required.

Designing Your SA Relationship Types and Service Providers

As explain in [A Service Agreement Can Have Many Types Of Relationships](#), SA relationship types and service providers are only required when you subcategorize your service agreements.

The topics in this section describe how to design your SA relationship types.

Designing SA Relationship Types

The easiest way to design SA relationship types is to start with the matrix of SA types designed in [SA Types And The Financial Design](#). For each SA type in the matrix, determine if either of the following questions is true:

- Can companies other than your own provide some subcategory of the service (and do you have some type of interaction with these companies)? For example, this would be true if you are an energy supply company because a different company is responsible for distributing the power to the customer (and you probably exchange consumption and financial transactions with this company).
- Does your organization use different rates for subcategories of the service? For example, this would be true if you use separate rates for water and wastewater service (even though both are based on the customer's water consumption).

If either of the above is true, you will need a SA relationship type for each subcategory of service.

We'll use an example to help make the point. Using the SA types we designed earlier (see [SA Types And The Financial Design](#)), we'll assume the following:

- We are designing the system for an electric, gas, water, waste water, and cable utility.
- Residential electric customers have a choice of energy supplier.
- Commercial and industrial electric customers can choose an energy supplier and a meter service provider.

- Commercial and industrial gas customers have a choice of energy supplier.

NOTE:

We are not showing most of the SA types that we designed earlier because they do not have subcategories of service.

CIS BU/ SA Type	SA Relationship Type
G/COM	Energy supply
G/IND	Energy supply
E/RES	Energy supply
E/COM	Energy supply
	Meter service
E/IND	Energy supply
	Meter service

NOTE:

Notice that we did not design a SA relationship type for our own distribution service. This is because our relationship type is implied (e.g., if you are a distribution company, you do not have to set up a SA relationship type for distribution service because the customer's "master" service agreement is implicitly associated with distribution service).

Designing Service Providers

After you design your SA relationship types, you need to list every potential service provider for each SA relationship type.

CIS BU/ SA Type	SA Relationship Type	Service Provider
G/COM	Energy supply	AmeriGas
		TransGas
		Green Power
G/IND	Energy supply	AmeriGas
		TransGas
		Green Power
E/RES	Energy supply	Green Power
		Cheap Power
		Us
E/COM	Energy supply	ElectriCorp
		Cheap Power
		TeniCorp
	Meter service	MeterCorp
		Us
E/IND	Energy supply	ElectriCorp

	Cheap Power
	TeniCorp
Meter service	MeterCorp
	Us

Next, list each unique service provider identified above:

Service Provider
AmeriGas
TransGas
Green Power
Cheap Power
Us
ElectriCorp
TeniCorp
MeterCorp

You will have at least one service provider for each entry in the above list. However, you may have to set up more than one service provider in the system for a given company. The topics below explain how this happens.

Billing Relationship Segmentation

As described under [Billing Relationships](#), a service provider may bill for you, you may bill for them, or you may each send a separate bill to the customer. In the table below, we have shown the assumed billing relationships for each service provider.

Service Provider	Billing Relationship
AmeriGas	We Bill For Them - Bill Ready
TransGas	We Bill For Them - Bill Ready
Green Power	We Bill For Them - Rate Ready
Cheap Power - Res	We Bill For Them - Rate Ready
Cheap Power - Com/Ind	Dual Billing
Us - Billable	It's Us
Us - Non billable	None
ElectriCorp	They Bill For Us - Bill Ready
TeniCorp	We Bill For Them - Bill Ready
MeterCorp	Dual Billing

Notice that we had to introduce additional service providers:

- Cheap Power has two service providers - one for residential customers, the other for commercial/industrial customers. This is necessary because we provide billing service for them for residential customers (We Bill For Them - Rate Ready), but for commercial and industrial customers they bill for themselves (Dual Billing).

- Our own service provider (the service provider known as "Us") has two service providers - one for energy supply because we bill for the energy we supply (It's Us), and another for meter service because we don't create bills for meter service (None).

We Bill For Them - Payment Relationship Segmentation

As described under [Pay At Bill Time vs. Pay At Pay Time](#) when you provide billing service for a service provider you have to define if you pay the service provider when you bill the customer OR only later, when the customer pays you.

In the table below, we have shown the payment relationships for each service provider.

Service Provider	Billing Relationship	Payment Relationship
AmeriGas	We Bill For Them - Bill Ready	Pay At Billing Time
TransGas	We Bill For Them - Bill Ready	Pay At Billing Time
Green Power	We Bill For Them - Rate Ready	Pay At Billing Time
Cheap Power - Res	We Bill For Them - Rate Ready	Pay At Billing Time
Cheap Power - Com/Ind	Dual Billing	N/A
Us - Billable	It's Us	N/A
Us - Non billable	None	N/A
ElectriCorp	They Bill For Us - Bill Ready	N/A
TeniCorp - Com	We Bill For Them - Bill Ready	Pay At Billing Time
TeniCorp - Ind	We Bill For Them - Bill Ready	Pay At Pay Time
MeterCorp	Dual Billing	N/A

Notice that we had to introduce an additional service provider for TeniCorp because for commercial customer we purchase the receivable (Pay At Bill Time), but for industrial customers we only pay them when we're paid by the customer (Pay At Pay Time).

Consumption Relationship Segmentation

As described under [Consumption Relationships](#) a service provider may send the customers' consumption to you, you may send consumption to them, or you may have no consumption relationship with a given service provider.

In the table below, we have shown the consumption relationships for each service provider.

Service Provider	Billing Relationship	Payment Relationship	Consumption Relationship
AmeriGas	We Bill For Them - Bill Ready	Pay At Billing Time	We Send Consumption
TransGas	We Bill For Them - Bill Ready	Pay At Billing Time	We Send Consumption
Green Power	We Bill For Them - Rate Ready	Pay At Billing Time	N/A
Cheap Power - Res	We Bill For Them - Rate Ready	Pay At Billing Time	N/A
Cheap Power - Com/Ind	Dual Billing	N/A	N/A
Us - Billable	It's Us	N/A	N/A
Us - Non billable	None	N/A	N/A
ElectriCorp	They Bill For Us - Bill Ready	N/A	N/A
TeniCorp - Com	We Bill For Them - Bill Ready	Pay At Billing Time	We Send Consumption

TeniCorp - Ind	We Bill For Them - Bill Ready	Pay At Pay Time	We Send Consumption
MeterCorp	Dual Billing	N/A	N/A

Notice that we didn't have to proliferate service providers due to consumption relationships.

Other Segmentations

The earlier parts of this discussion described the most common factors that cause the creation of service providers. However, many obscure factors could cause the introduction of more service providers. In this section we explain these more obscure factors.

Payment Method

As described under [Pay At Bill Time vs. Pay At Pay Time](#), when you provide billing service for a service provider you have to pay the service provider at some point in time. The algorithm that defines the amount to pay (and how the related adjustment is generated) is defined on the service provider record. If a service provider has different payment algorithms for different customer segments, you must split the service provider accordingly.

Transfer Receivable Method

As described under [When They Bill For Us, They Owe Us Money](#), a service provider will owe you money if they provide billing service for you. The algorithm that defines how to transfer the customer's receivable to the service provider is defined on the service provider record. If a service provider has different transfer A/R algorithms for different customer segments, you must split the service provider accordingly.

Billable Charge Download Process

As described under [They Bill For Us - Bill Ready](#), billable charges are interfaced to service providers who provide billing service for you. The background process that performs the interface of billable charges is defined on the service provider record. If a service provider has different billable charge interfaces for different customer segments, you must split the service provider accordingly.

Consumption Download Process

As described under [We Can Send Billed Consumption To Any Service Provider](#), consumption can be sent to any service provider. The background process that performs the interface of consumption is defined on the service provider record. If a service provider has different consumption interfaces, you must split the service provider accordingly.

Notification Upload Processing

As described under [Designing Notification Upload & Workflow Procedures](#), a service provider can send you notifications. Whenever a notification is uploaded, the system creates a workflow process to process each such notification. The type of workflow process that's created is controlled by the service provider's workflow process profile. If a service provider

requires a different workflow process for a given type of notification (for whatever reason), you must split the service provider accordingly.

Notification Download Processing

As described under [Designing Notification Downloads](#), the system will send notifications to service providers when something noteworthy happens and when information is needed from a service provider. The type of notification that is sent to a service provider and the background process that interfaces the notification to the service provider is defined on the service provider's notification download profile. If a service provider requires a different type of notification to be sent or they have different interface protocols for a given type of notification, you must split the service provider accordingly.

Financial Settlement Service Agreement

As described under [Service Providers Have Service Agreements Too](#), service providers have service agreements. These service agreements contain how much you owe the service provider (if you bill for them) and how much they owe you (if they bill for you). If you want to have separate service agreements for financial settlements associated with distinct customer segments, you must split the service provider accordingly.

Geographic Area Segmentation

The following table reflects the new service providers that were added since we started [Designing Service Providers](#).

CIS BU/ SA Type	SA Relationship Type	Service Provider
G/COM	Energy supply	AmeriGas
		TransGas
		Green Power
G/IND	Energy supply	AmeriGas
		TransGas
		Green Power
E/RES	Energy supply	Green Power
		Cheap Power - Res
		Us - Billable
E/COM	Energy supply	ElectriCorp
		Cheap Power - Com/Ind
		TeniCorp - Com
	Meter service	MeterCorp
	Us - Non billable	
E/IND	Energy supply	ElectriCorp
		Cheap Power - Com/Ind
		TeniCorp - Ind
	Meter service	MeterCorp

Next, determine the postal code ranges in which a service provider is allowed to provide service.

CIS BU/ SA Type	SA Relationship Type	Service Provider	Postal Range
G/COM	Energy supply	AmeriGas	94000 - 95999
		TransGas	94000 - 95999
		Green Power	94000 - 95999
G/IND	Energy supply	AmeriGas	94000 - 95999
		TransGas	94000 - 95999
		Green Power	94000 - 95999
E/RES	Energy supply	Green Power	94000 - 95999
			93000 - 93999
		Cheap Power - Res	94000 - 95999
		Us - Billable	94000 - 95999
E/COM	Energy supply	ElectriCorp	94000 - 95999
		Cheap Power - Com/Ind	94000 - 95999
		TeniCorp - Com	94000 - 95999
	Meter service	MeterCorp	94000 - 95999
		Us - Non billable	94000 - 95999
E/IND	Energy supply	ElectriCorp	94000 - 95999
		Cheap Power - Com/Ind	94000 - 95999
		TeniCorp - Ind	94000 - 95999
	Meter service	MeterCorp	94000 - 95999
		Us - Non billable	94000 - 95999

Next, we need to strip off the SA types because the postal ranges are defined for combinations of service provider and SA relationship type. Notice the problem - we have a service provider - Green Power has different postal ranges for the same SA relationship type. You have two ways to fix this problem, you can split your service provider (have one for the gas and another for the electric), or you can split the SA relationship type (have one for the gas and another for the electric). We've chosen the former in our example.

SA Relationship Type	Service Provider	Postal Range
Energy supply	AmeriGas	94000 - 95999
	TransGas	94000 - 95999
	Green Power - Gas	94000 - 95999
Energy supply	AmeriGas	94000 - 95999
	TransGas	94000 - 95999
	Green Power - Gas	94000 - 95999
Energy supply	Green Power - Electric	94000 - 95999
		93000 - 93999

	Cheap Power - Res	94000 - 95999
	Us - Billable	94000 - 95999
Energy supply	ElectriCorp	94000 - 95999
	Cheap Power - Com/Ind	94000 - 95999
	TeniCorp - Com	94000 - 95999
Meter service	MeterCorp	94000 - 95999
	Us - Non billable	94000 - 95999
Energy supply	ElectriCorp	94000 - 95999
	Cheap Power - Com/Ind	94000 - 95999
	TeniCorp - Ind	94000 - 95999
Meter service	MeterCorp	94000 - 95999
	Us - Non billable	94000 - 95999

In the table below, we have shown the final list of service providers.

Service Provider	Billing Relationship	Payment Relationship	Consumption Relationship
AmeriGas	We Bill For Them - Bill Ready	Pay At Billing Time	We Send Consumption
TransGas	We Bill For Them - Bill Ready	Pay At Billing Time	We Send Consumption
Green Power - Electric	We Bill For Them - Rate Ready	Pay At Billing Time	N/A
Green Power - Gas	We Bill For Them - Rate Ready	Pay At Billing Time	N/A
Cheap Power - Res	We Bill For Them - Rate Ready	Pay At Billing Time	N/A
Cheap Power - Com/Ind	Dual Billing	N/A	N/A
Us - Billable	It's Us	N/A	N/A
Us - Non billable	None	N/A	N/A
ElectriCorp	They Bill For Us - Bill Ready	N/A	N/A
TeniCorp - Com	We Bill For Them - Bill Ready	Pay At Billing Time	We Send Consumption
TeniCorp - Ind	We Bill For Them - Bill Ready	Pay At Pay Time	We Send Consumption
MeterCorp	Dual Billing	N/A	N/A

Designing Your SA Types And Start Options For Sub SAs

When you were [Designing Service Providers](#), you defined the service providers that were valid for every combination of SA type and SA relationship type. If you provide billing services for another service provider or if you subcategorize your own services, you another task - you have to design the SA types for your sub SAs.

As described earlier, there will be a separate [Sub Service Agreement](#) for every SA relationship for which we calculate a bill segment. Every sub SA must reference an SA type. The following table shows sample SA types (notice that they are only used for We Bill For Them and It's Us service providers).

CIS BU/ SA Type	SA Relationship Type	Service Provider	SA Type(s) for Sub SAs
G/COM	Energy supply	AmeriGas	AG1

		TransGas	TG1
		Green Power - Gas	GP-GC1
G/IND	Energy supply	AmeriGas	AG1
		TransGas	TG1
		Green Power - Gas	GP-GI1
E/RES	Energy supply	Green Power - Electric	GP-ER1
		Cheap Power - Res	CP-ER1
		Us - Billable	US-ER1
E/COM	Energy supply	ElectriCorp	Not applicable - they bill for us
		Cheap Power - Com/Ind	Not applicable - dual billing
		TeniCorp - Com	TC1
	Meter service	MeterCorp	Not applicable - dual billing
		Us - Non billable	Not applicable - no billing
E/IND	Energy supply	ElectriCorp	Not applicable - they bill for us
		Cheap Power - Com/Ind	Not applicable - dual billing
		TeniCorp - Ind	TC1
	Meter service	MeterCorp	Not applicable - dual billing
		Us - Non billable	Not applicable - no billing

The design steps required to set up these SA types are similar to those described under [Designing SA Types For Service Agreements With Service Points](#). The following points provide a few suggestions that will help you design your SA types for sub SAs:

- The business unit should be the same as defined for the master service agreement.
- Service type should be set up as per [Service Segmentation](#).
- Distribution code should be set up as per [Receivable Segmentation](#).
- Obviously, the sub SA switch should be turned on.
- Start options:
 - If the SA type is used for a We Bill For Them - Rate Ready service provider or for yourself, the start options should be Required because rates, contract riders and/or contract values will be populated on the sub SA from a start option. Refer to [Automatic Creation of Sub SAs](#) for more information.
 - If the SA type is used for a We Bill For Them - Bill Ready service provider, the start options should be not allowed because we don't need to default rates, contract riders and/or contract values on billable charge sub SAs.
- The payment distribution and late payment information should be set up as for any SA type. Refer to [Cash Distribution Segmentation](#) and [Late Payment Charge Segmentation](#) for more information.
- If the SA type is used for We Bill For Them - Bill Ready service provider, the special role should be Billable Charge, otherwise it should not be used. Remember, you should not use sub SAs for Cash Deposits.
- Deposit class should be used if the sub SA is covered by a deposit. Refer to [Deposits Issues](#) for more information.
- The one time charge switch should be off.
- Bill segment type:
 - If the SA type is used for a We Bill For Them - Bill Ready service provider, the bill segment type should reference a bill segment creation algorithm that creates bill segments from billable charges (and generates bill segment errors until the last night of the bill cycle). Refer to [The Bill Ready Calculation Method](#) for more information.

- If the SA type is used for a We Bill For Them - Rate Ready service provider, the bill segment type should reference a get consumption algorithm that gets consumption from the master SA, and a bill segment creation algorithm that applies a rate.
- Specify a characteristic premise is required if the sub SA is associated with premise-oriented service.
- The calendar billing options should not be used.
- The recurring charge information should not be used.
- Sub SAs used for We Bill For Them - Rate Ready service providers should NOT be eligible for budgets and therefore the eligible for budget switch should be off. This admonition is given because budget billing causes current amount due to be out-of-sync with payoff amount due and we don't want this to happen for sub SAs. Why? Because we use the bill segments associated with these sub SAs to determine how much we owe the service provider.
- If the SA type is used for a We Bill For Them - Rate Ready service provider or for your own company, rates need to be set up; otherwise they should not be used.
- Sub SAs never reference service points. Refer to [Only The "Master" Service Agreement Is Linked To Service Points](#) for more information.
- Adjustment types profiles should be set up accordingly.
- Refer to [Credit and Collection Issues](#) for recommendations in respect of the debt class, write off debt class, and severance criteria associated with these SA types.
- Billable charge templates should not be used.
- Completion algorithms cannot be used for sub SAs.
- If the SA type is used for a We Bill For Them service provider, you should link to the SA type the FT freeze algorithm that controls how we pay the service provider. Refer to [Pay At Bill Time vs. Pay At Pay Time](#) for how this algorithm is used.
- The creation of bill segments for the sub SAs occurs after the bill segment for the related master SA is created. If you populate a [billing processing sequence](#) on an SA type for a sub SA, it is used to control the order in which the sub SAs for a given master SA are processed relative to each other.

Refer to [Setting Up SA Types](#) for how to set up these new SA types in the system.

Reference Send Consumption Algorithm On Master SA Types

As explained under [We Can Send Billed Consumption To Any Service Provider](#), when a master SA's bill segment is frozen, the system must check if there are any service providers who need the bill segment's consumption. If so, it sets up the data necessary to interface the master SA's consumption (snapshot on the bill segment) to the service provider(s). The system will only do this if you specify an appropriate FT Freeze Algorithm on the master SA types. Refer to [SA Type - Algorithm](#) (FT Freeze Algorithm) for more information.

Reference TBFU Algorithm On Master SA Types

As explained under [They Bill For Us](#), when a bill is completed, the system needs to check if there are any service providers who bill for us associated with the bills "master" SAs. If so, it sets up the data necessary to interface the master SA's charges to the service provider and to transfer the receivable balance from the customer to the service provider. The system will only do this if you specify an appropriate FT Completion Algorithm on the master SA types. Refer to [SA Type - Algorithm](#) (Bill Completion Algorithm) for more information.

NOTE:

If there are multiple master SAs on a bill, the financial transactions associated with each respective master SA could be routed to different service providers (e.g., one service provider could bill for gas and another could bill for electricity). Refer to [Different Service Providers Can Bill Different Services](#) for more information.

Designing SA Types For Service Provider Financial Settlements

As explained in [Service Providers Have Service Agreements Too, We Bill For Them and They Bill For Us](#) service providers require a service agreement. You must create SA types for these types of service agreements. The following points provide a few suggestions that will help you design these financial settlement SA types:

- Service type should probably be a non-service oriented service type.
- Distribution code for We Bill For Them settlement SAs should be a payable account (or treat it as a "contra" receivable). Refer to [Receivable Segmentation](#) for more information.
- The sub SA switch should be turned off.
- Start options are not allowed.
- The payment distribution and late payment information should be set up as for any SA type. Refer to [Cash Distribution Segmentation](#) and [Late Payment Charge Segmentation](#) for more information.
- Special role should not be used.
- Deposit class should be used if the settlement service agreement is covered by a deposit. This would probably only be used for They Bill For Us service providers (because they will owe us money).
- The one time charge switch should be off.
- These service agreements are not billable and therefore none of the billing information should be specified.
- The characteristic premise switch should be off.
- Rates should not be used.
- Service points should not be used.
- Adjustment types profiles should be set up accordingly.
- Debt class, write off debt class, and severance criteria should be set up accordingly.
- Billable charge templates should not be used.
- Completion algorithms should not be used.
- Freeze algorithms should not be used.

Refer to [Setting Up SA Types](#) for how to set up these new SA types in the system.

Setting Up SA Relationship Information

In the previous section, [Designing Your SA Relationship Types and Service Providers](#), we presented a case study that illustrated a mythical organization's SA relationship information. In this section we explain how to set up this information.

Setting Up SA Relationship Types

Open **Admin** > **Open Market** > **SA Relationship Type** to define your SA relationship types. Refer to [Designing SA Relationship Types](#) for more information.

Description of Page

Enter an **SA Relationship Type** code and **Description** for every relationship type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SA_REL_TYPE](#).

Setting Up Service Providers

The topics in this section describe how to set up service providers.

Service Provider - Main

Open **Admin > Open Market > Service Provider > Add** to define core information about a service provider.

Description of Page

Enter a unique **Service Provider** code for the service provider.

End a brief **Description** of the service provider.

If you communicate with this service provider via notification messages or outbound messages, indicate the service provider's **External System**.

If you send notifications to this service provider, select a **Notification DL (download) Profile** that is used to define the configuration of the outgoing messages. Refer to [Designing Notification Download Profiles](#) for more information.

Select the **Person ID** that contains this service provider's phone numbers and demographic information.

If you bill for the service provider or if they bill for you, select the **Service Agreement** that holds how much you owe them or they owe you. Refer to [Service Providers Have Service Agreements Too](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SPR](#).

Service Provider - Detail

Open **Admin > Open Market > Service Provider > Search** and navigate to the **Detail** page to define additional information about your service providers.

Description of Page

Define the **Billing Relationship** you have with the service provider. Refer to [Billing Relationships](#) for more information.

If you provide billing services for this service provider (i.e., the **Billing Relationship** is We Bill For Them), define the **Payment Relationship**. Refer to [Pay At Bill Time vs. Pay At Pay Time](#) for more information. You may not be paying some service providers as such. Rather, the customer's receivables are simply transferred to the service provider, e.g., when you calculate discounts for special [negotiated terms](#). For these service providers, choose a payment relationship of Pay SP Not Applicable.

Define the **Consumption Relationship** you have with the service provider. Refer to [Consumption Relationships](#) for more information.

If a service provider reads meters and they can override the customer's service cycle, turn on **Overrides ServiceCycle**. Refer to [MDMAs And Service Cycles](#) for more information.

As described under [Pay At Bill Time vs. Pay At Pay Time](#), when you provide billing service for a service provider you have to pay the service provider at some point in time. The **Pay Service Provider Algorithm** defines the amount to pay and how

the related adjustment is generated. Refer to [Paying The Service Provider - Technical Implementation](#) for more information about how this algorithm is used. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that increases how much is owed the service provider. Click [here](#) to see the algorithm types available for this plug-in spot.

As described under [When They Bill For Us, They Owe Us Money](#), a service provider will owe you money if they provide billing service for you. The **Transfer A/R Algorithm** defines how to transfer the customer's receivable to the service provider. Refer to [A/R Transfer - Technical Implementation](#) for more information about how this algorithm is used. This algorithm is also used to transfer receivables when you calculate discounts for special [negotiated terms](#). If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that transfers financial transactions from the customer to the service provider. Click [here](#) to see the algorithm types available for this system event.

As described under [They Bill For Us - Bill Ready](#), billable charges are interfaced to service providers who provide billing service for you. The **Billable Charge Download Process** performs the interface of billable charges to the service provider. Refer to [Routing Billable Charges To Service Providers - Technical Implementation](#) for more information about how this process is used.

As described under [We Can Send Billed Consumption To Any Service Provider](#), consumption can be sent to any service provider. The **Consumption Download Process** performs the interface of consumption to the service provider. Refer to [Routing Consumption To Service Providers - Technical Implementation](#) for more information about how this process is used.

Service Provider - SA Relationship Type

Open **Admin > Open Market > Service Provider > Search** and navigate to the **SA Relationship Type** page to define the types of relationships (e.g., energy supplier, energy distributor, meter data management agency) associated with a service provider and the postal code ranges in which the service provider operates.

Description of Page

Use the **SA Relationship Types** collection to define this service provider's **SA Relationship Types**. Use the collection that appears in the grid to define the **Postal Code** ranges in which this service provider is allowed to operate for each **SA Relationship Type**.

IMPORTANT:

After defining the SA relationship types that can be associated with a service provider, you must then define the SA types on which the service provider / SA relationship type combination can be used. This information is defined using [Setting Up SA Relationships For SA Types](#).

Service Provider - Bill Messages

Open **Admin > Open Market > Service Provider > Search** and navigate to the **Bill Messages** page to define bill messages to appear on bills that contain charges associated with a service provider.

Description of Page

Use the **Bill Messages** collection to define **Bill Message** codes that should appear on bills that contain charges associated with a given service provider. For each message, also specify the **Start Date** and **End Date** when such a message should appear on the bill (leave **End Date** blank if the message should appear indefinitely).

Where Used

The system snaps bill messages on a bill during bill completion. Refer to [The Source Of Bill Messages](#) for more information.

Setting Up SA Types and Start Options For Sub SAs

The SA types and start options described under [Designing Your SA Types And Start Options For Sub SAs](#) must be set up. Refer to [Setting Up SA Types](#) for how to do this.

Setting Up SA Types For Financial Settlements

The SA types described under [Designing SA Types For Service Provider Financial Settlements](#) must be set up. Refer to [Setting Up SA Types](#) for how to do this.

Update Master SA Types With FT Freeze and Bill Completion Algorithms

Refer to [SA Type - Algorithm](#) for more information.

Setting Up SA Relationships For SA Types

SA Type SA Relationship Type - Main

Open **Admin** > **Open Market** > **SA Type SA Rel. Type** > **Add** to define the types of SA relationships and service providers that can be associated with a SA type.

Description of Page

Define the **SA Relationship Type** that can be associated with service agreements of this **SA Type**.

NOTE:

You may only define SA Relationship Types for "master service agreements".

Turn on the **Required** switch if this **SA Relationship Type** must be defined for service agreements of this type. Refer to [Defaulting Relationship Types And Defaulting Service Providers](#) and [Required Relationship Types and Billing](#) for more information.

Indicate if **Gaps in SA Relationships** of this type that are associated with this service agreement type are Allowed or Not Allowed. You should only select Allowed if relationships of this type can be expired without a relationship with another service provider to replace it. Deregulated relationships typically should not have gaps in the relationship. For example, a relationship with an energy service provider should not expire unless a relationship with another energy service provider replaces it. Refer to [negotiated terms](#) for an example of SA relationship types that allow gaps in SA relationships.

Use the collection to define the **Service Providers** who can be associated with this **SA Relationship Type** on service agreements of this **SA Type**.

NOTE:

Only **Service Providers** previously defined as being valid for the **SA Relationship Type** can be specified (refer to [Service Provider - SA Relationship Type](#) for how to link a service provider to a SA relationship type).

Turn on **Default SPR** if the **Service Provider** should be defaulted on newly created SA relationships. Refer to [Defaulting Relationship Types And Defaulting Service Providers](#) for more information.

Use **Status** to control if the **Service Provider's** relationship to the SA Type / SA Relationship type is Active or Inactive. Only Active service providers can be linked to service agreements of this type. The system allows Inactive service providers in order to support historical service providers who are no longer active and to allow you to set up new service providers in advance of their start date.

Use the drill down button adjacent to a service provider to view the valid Sub SA Types. Alternatively, navigate the Sub SA Type tab and scroll until you find the desired service provider.

Where Used

When a new SA relationship is defined for a service agreement, the system uses this information to make sure the relationship is valid and that the associated service provider is valid.

SA Type SA Relationship Type - Sub SA Type

Open **Admin > Customer > SA Type SA Rel Type > Search** and navigate to the **Sub SA Type** page to define valid sub SA types for service providers associated with a SA type.

If you provide billing services for the service provider (i.e., the service provider's billing relationship is We Bill For Them) or if you subcategorize your own charges (i.e., the service provider is your organization and it has a billing relationship of It's Us) a [Sub Service Agreement](#) will be created for the service provider.

Description of Page

The information in the collection defines the valid **Sub CIS Division** and **Sub SA Types** of these sub service agreements. Those entries marked as **Create Initially** are used by the process that creates sub SAs for new SA relationships. This background process uses this information as follows:

- If the sub SA's SA type doesn't use Start Options (as defined on SA Type - Main), the background process simply creates a sub SA with the given SA type. Note: these types of sub SAs are typically used for service providers who send their charges to you (i.e., they have a billing relationship of We Bill For Them - Bill Ready). This is because Billable Charge service agreements are used for these types of service agreements and billable charge service agreements contain very little information.
- If the sub SA's SA type uses Start Options (as defined on SA Type - Main), the **Start Option** defined in the collection is used to populate the sub SA with default values (e.g., rate, contract rider, etc.). Refer to [Setting Up Start Options](#) for more information.

Where Used

The process that creates sub SAs for new SA relationships uses this information to determine the number and type of sub SAs to create for each SA relationship.

Negotiated Terms

The topics in this section describe the use of SA relationship functionality for applying certain types of negotiated terms. It assumes that you are familiar with SA relationship functionality in general.

NOTE:

Negotiated terms are optional. The functionality described in this section is only relevant if your organization offers this functionality.

Umbrella agreements. If the Contract Management module is not **turned off**, you may also choose to use umbrella agreements to manage the functionality described here.

Imagine that the head office for a multi-site organization negotiates special terms that cover a number of its sites, each of whom have their own account and service agreement. The negotiated terms typically involve discounts. These discounts may be realized

- Under a single service agreement, separate and distinct from the service agreements that are covered, or
- Individually, for each covered service agreement. In this case, two sub-scenarios exist:
 - The discounts may be reflected on the bill segment of each service agreement, or
 - The discounts may be transferred to another service agreement (the group's service agreement). The discount does NOT appear on the individual bill segment for the service agreements that are covered by the negotiated terms.

SA relationships track and manage complex business relationships between a customer and a service provider. You can define the above relationships using service agreement relationship functionality. This is a special type of SA relationship in which the head office is the service provider.

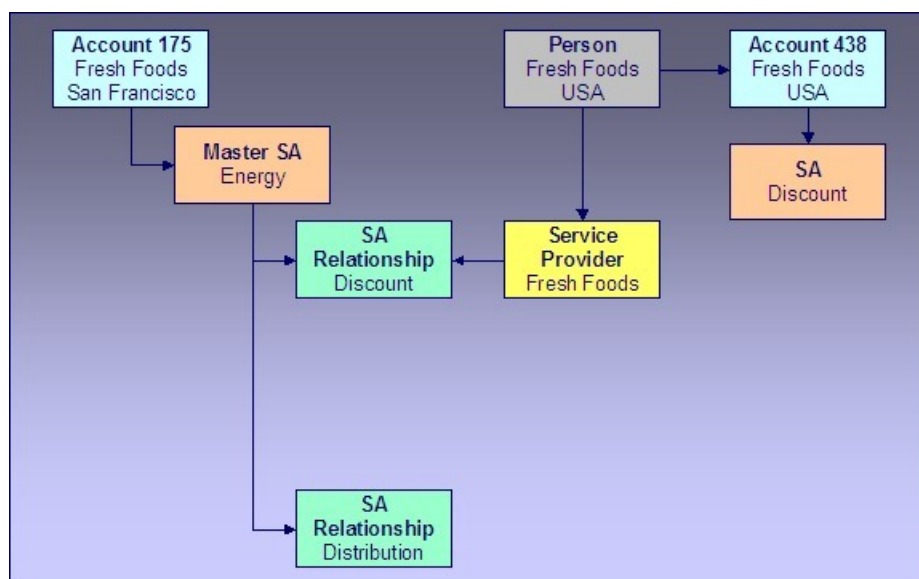
Examples Of Special Discounts

The following topics illustrate examples of how your organization may configure the system to handle this business functionality.

Example Using Aggregated Consumption

A customer, Fresh Foods has 10 stores each with its own account and electric service agreement, i.e., each store is billed separately. Fresh Food's head office decides to negotiate a group discount that applies to one or more of the individual stores.

In the following diagram, one store's account is shown.



Example Using Aggregated Consumption

Note the following:

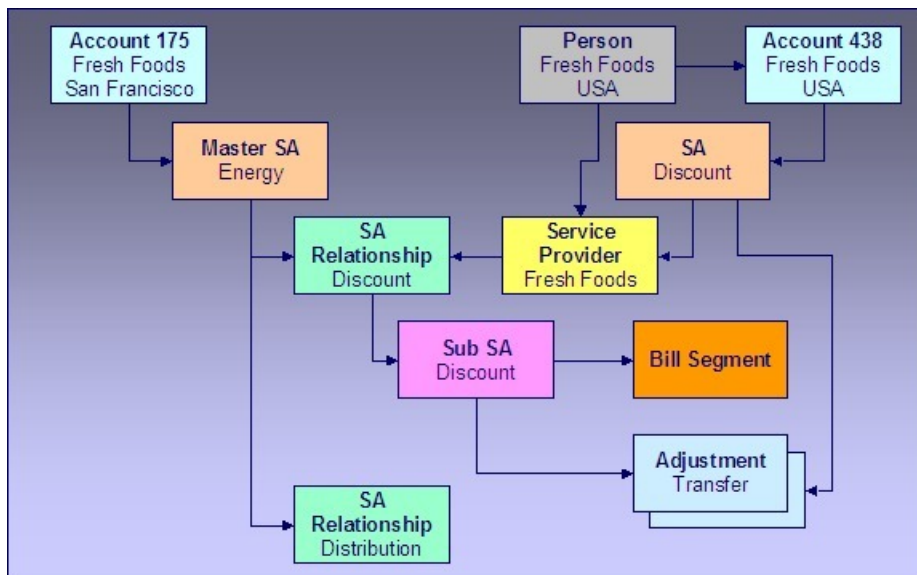
- An SA relationship is created for the energy SA covered by the negotiated terms.
- The service provider in the relationship is the head office.
- This relationship does not have a sub-SA because no additional billing services are provided for each individual covered service agreement.
- Discounts are calculated using the head office discount service agreement and affect only this service agreement and therefore only the head office's account.
- The master SA may have other deregulated relationships, such as the distribution relationship shown.

NOTE:

Aggregated consumption algorithm. In this scenario, the service provider's service agreement amalgamates the group's consumption and applies a rate to calculate the discount. This algorithm is not provided - you get to write this algorithm to meet your needs.

Example Using Site-by-site Discount

In this scenario, the discount is calculated on a site-by-site basis.



Example Using Site By Site Discount

Note the following differences between this example and the one where consumption is aggregated:

- Each service agreement participating in the discount has an SA relationship with a sub SA.
- Discounts are calculated on a site-by-site basis by the sub SA for each service agreement covered by the discount agreement.
- The discounts may be transferred to the head office service agreement using transfer adjustments. In this case, you may choose to not show the discount on the individual site's bills.

Setting Up The System To Enable Negotiated Terms

The above topics provided background information about how special negotiated terms could be supported in the system. The following discussion summarizes the various setup tasks alluded to above. These notes highlight the setup required in addition to that detailed in [Setting Up SA Relationships Information](#).

Algorithm

If you use sub SAs to calculate charges / discounts for each covered service agreement individually and you want to transfer the charges / discounts to the service provider's service agreement at bill completion, you will need to set up a bill completion algorithm to do this.

Refer to the algorithm type [BCMP-TR](#) for more information.

SA Relationship Type

You will need an SA relationship type that will be used for each type of negotiated term or discount.

Service Provider

Set up a service provider using the principal party of the negotiated discount. For example, if a head office negotiates a special agreement for its sites, you should set up a new service provider representing the head office.

- **Billing Relationship** will depend on the type of negotiated terms you choose to set up. For terms where debits / credits are calculated for each covered service agreement individually, select We Bill For Them, Rate Ready. We are effectively billing on behalf of the principal party. For terms where the discount is calculated on an amalgamated SA there is no billing relationship because no additional billing services are provided for each covered service agreement.
- Select a **Payment Relationship** of Pay SPR Not Applicable if debits / credits to the sub-SA should not be transferred to the service provider. If you want to transfer charges at bill completion time using a bill completion algorithm, you should also select Pay SPR Not Applicable because you will not be creating adjustments to pay the principal party service at payment time or at billing time.
- Enter a **Transfer A/R Algorithm** to transfer the debit / credit from the sub-SA to the principal party's service agreement. If you do not want to transfer the debit / credit to the principal party's service agreement, you do not need to specify a transfer algorithm.

SA Type SA Relationship Type

Associate the SA relationship type setup for negotiated terms with the SA type(s) of the service agreements that are covered by the terms. The following are suggested values for the SA type SA relationship type:

- **Required.** Relationships of this type should not be required for the SA type because the negotiated terms will only cover selected service agreements with that SA type.
- **Gaps in SA Relationship** are Allowed if service agreements of this type do not have to have an SA relationship of this type with a service provider throughout the effective period of the SA.
- Add the service provider set up above to the list of valid service providers for this SA type and SA relationship type combination.

SA Type

Set up an SA type for the sub SAs that will be used to calculate discounts. This SA type should use the [BCMP-TR](#) bill completion algorithm if you want to transfer the discounts to the head office.

NOTE:

TBFU Deregulated Relationship with WBFT Negotiated Terms Relationship. If you have the unusual situation where a master service agreement has a relationship with a TBFU service provider as well as a negotiated term "relationship" with a WBFT service provider, and you transfer charges / discounts from the discount agreement sub SA to the WBFT service provider, you will not be able to use the algorithms provided to transfer receivables to the service providers. This is because bill completion algorithms associated with the SA type of a master service agreement are executed before any bill completion algorithms associated with the SA types of sub service agreements. Consequently the TBFU XFER algorithm associated with the master service agreement will transfer all receivables from related sub SAs to the TBFU service provider before the BCMP-TR algorithm can transfer the discount agreement receivables. You will need to modify the TBFU XFER bill completion algorithm that transfers receivables from the customer to the TBFU service provider to exclude the discount agreement receivables.

Defining Service Credit Options

Some companies allow their customers to participate in a special rewards program. The term "service credits" is used to describe a program that rewards customers for their business. The topics in this section provide details to help you set up the control tables required to support any service credit program that your company supports.

NOTE:

Penalty Points. The service credits functionality is described in the documentation with the assumption that it is used for accumulating points to reward your customers. However, if your company has a business need to record penalty points for a customer, the service credits functionality may be used for that purpose as well.

The Big Picture Of Service Credits

The topics in this section provide background information about service credit functionality.

Service Credit Membership

Let's look at some examples of special programs that may use service credits functionality:

- Capital credits - When customers receive their utility service from a cooperative, they are considered "members" of the cooperative and may over time receive capital credit allocations from the cooperative based on their service history and the cooperative's profits allocated during that time period.
- Frequent flier miles - Perhaps your company has made an agreement with one or more airlines to allow customers to accumulate frequent flier miles for every x amount spent on service.

- Free pay-per-view movies - Maybe your cable service offers free pay-per-view movies under certain conditions. Using service credits, you can set up your system to accumulate the free pay-per-view movies and use the free movies to offset actual movies viewed by the customer.
- Any other type of loyalty program where the customer earns credits that may later be redeemed in some way.

To participate in a program such as those described above, the customer is linked to a service credit membership. The membership record provides the following functionality:

- It defines the accounts that are linked to the membership.

FASTPATH:

Refer to [Who are the Members?](#) for information about linking persons and accounts to a membership.

- It defines a [membership type](#), which controls certain behavior about the membership.
- It may define an external ID if the membership is associated with an external program, such as a frequent flier mile program.
- It may define a [service agreement](#) to use for miscellaneous financial transactions that may get created.
- It may define characteristics used to capture miscellaneous information about the membership.
- Over time, service credit events are created for a membership. The events indicate an amount that either adds or subtracts credit units (i.e., points, miles, movies, dollars, etc) for the membership.

FASTPATH:

Refer to [The Big Picture of Service Credit Membership](#) for more information about functionality related to a membership.

How Are Service Credits Earned?

Service credits may be monetary rewards for service or they may be non-monetary rewards such as free movies or frequent flier miles. In any case, how the membership earns the points or rewards depends on the business rules for the program you are offering.

A typical scenario is that the service credits are earned for a membership as a result of other services linked to the membership's accounts. For example:

- Perhaps free pay-per-view movies are earned when signing up for cable service. In this case, the pay-per-view movie membership is related to the membership account's cable service agreement. Refer to [Service Credits Earned When Starting Service](#) for more information.
- Perhaps one frequent flier mile is earned for every \$10 spent on electricity. In this case, the frequent flier membership is related to the membership account's electricity service agreements (for example, electricity distribution, electricity retail and lamp service). Refer to [service credits earned through billing](#) for more information.
- For a capital credits membership, capital credit allocations are calculated based on the amount spent by the customer for standard service, for example electricity and/or gas service. In this case, the capital credit membership is related to the membership account's electricity and/or gas service agreements. For capital credits, a background process is used to calculate the allocated amounts. Refer to [Allocating Capital Credit](#) for more information.

It is also possible to earn service credits irrespective of other service for the membership's accounts. (Again, it depends on the business rules for the program you are offering.) For example, perhaps you offer 500 frequent flier miles for signing up for service with your company, regardless of the type of service chosen. In addition, assume that no additional miles are earned for ongoing service. In this example, there is no need to link the membership to any service agreements.

FASTPATH:

Refer to [Service Agreements Contribute to a Membership](#) for more information.

Each earned service credit amount is linked to the membership via a [service credit event](#).

How Are Service Credits Redeemed?

Once service credits have been earned for a membership, how may a customer redeem these credits? The method by which the credits are redeemed depends on the business rules for the program you are offering. Here are some examples of how a service credit may be redeemed:

- For free pay-per-view movies, perhaps your customer's monthly cable bill is credited for any pay-per-view movies until all free movies are used. Refer to [Service Credits Redeemed Through Billing](#) for more information.
- For frequent flier miles, the information about the earned miles is exported to the appropriate airline. The miles are actually redeemed by the customer through the airline, not through your company. Refer to [Interface Membership Information to a Third Party](#) for more information.
- For a capital credit membership, the company periodically (maybe once a year) decides if credits should be redeemed (referred to as "retired") and if so, how much. The company runs a background process to calculate the "retirement" amount. When an amount is retired, the membership balance is reduced by the retirement amount and the amount is transferred to service agreements related to the membership. Refer to [Capital Credit Retirement](#) for more information.

NOTE:

Tracking Membership Balances. If service credits are redeemed via the system, your membership should probably be configured to keep track of a balance. Refer to [Event Amounts May Contribute to a Balance](#) for more information.

Each service credit amount redeemed through the system is linked to the membership via a [service credit event](#).

Service Agreements For a Membership

There are three types of service agreements that may be associated with a service credit membership. The following sections describe these types.

SAs Contribute to the Membership

As described in [how are service credits earned](#), many memberships are related to specific service agreements for the membership's accounts. We refer to these service agreements as the SAs that contribute to the membership because often the service credit amounts earned for the membership are based on amounts spent by the customer for these services.

During the lifetime of a membership, service agreements that contribute to a membership may be stopped and other service agreements started. For example, perhaps you have a frequent flier membership that is related to electric service. Imagine that the customer starts out with a certain rate for electric service, but later decides to opt for a different type of rate that requires expiring the old service agreement and creating a new one. How does this affect your frequent flier mile calculation? It depends on how you design the [algorithm](#) that creates the frequent flier events. Essentially, the algorithm must cater for this situation. The following diagram illustrates the scenario.

Service Credit Membership

Membership: 1231231123 – John Smith
 Type: Frequent Flier Miles for Continental
 Effective: 1/1/2000 -

Service Agreements

SA Type	SA ID	Start Date	End Date
E-RES	5487936555	1/1/2000	10/1/2000
E-RES	5487944872	10/1/2000	

Bill Generated: 09/15/2000 – 10/15/2000
 Bill Segment for 1st SA: 09/15/2000-10/1/2000
 Bill Segment for 2nd SA: 10/1/2000-10/15/2000

When calculating frequent flier miles, the algorithm should include the bill amount for both SAs for the given time period.

The service agreements that contribute to a membership are not linked directly to the membership record. This would cause a maintenance burden, requiring links to be updated when service is stopped or started for applicable service agreements. Rather, this link is indirect. The list of service agreement types is defined for the membership type. The system can determine which service agreements are "linked" to the membership by looking at the SA types for the membership type.

FASTPATH:

Refer to [Determine The Types of Service Agreements That Contribute to the Membership](#) for more information on designing your membership type to include appropriate SA types.

SA Used for Miscellaneous Transactions

For some memberships, you may need to define a special service agreement to use for miscellaneous transactions. For example:

If events created for a membership cause an adjustment to be created to affect the general ledger, rather than allowing the system to arbitrarily pick a service agreement to use for this adjustment, the SA to use should be indicated on the membership. Refer to [An Event May Cause Other Actions to Occur](#) for more information.

Membership Fee SAs

For some memberships, a [membership fee](#) may be applicable. A special service agreement is used to hold the fee. This service agreement is not linked directly to the membership, but is simply a service agreement linked to one of the membership's accounts.

NOTE:

Some fees may be refundable. The refunding of a fee must be handled by an algorithm. Refer to [SAST-RF](#) for information about the algorithm type provided with the base product.

Designing Your Service Credit Options

This section helps you to determine how to design your service credit membership types and service credit event types.

Designing Your Membership Types

This section discusses the options to consider when designing your service credit membership types.

First consider the type of unit that your membership's service credit events represent.

- Is it related to a currency? If so, ensure that your currency is correctly defined on the [currency](#) page.
- Is it a non-currency unit, such as movies, points or miles? If so, you need to define the unit on the [credit unit](#) page.

Next, consider other behavior that your membership may exhibit.

- Should your membership [calculate an overall balance](#) ?
- Will miscellaneous financial transactions be created for you membership over its lifetime? If so you may need to link a [service agreement to your membership](#).
- Should events linked to your membership reference a [fiscal year](#) ?

The table below illustrates three types of memberships: one for capital credits, one for frequent flier miles and one for free pay-per-view movies.

Membership Type	Description	Unit Type / Currency or Credit Unit	Has Balance?	Require SA?	Fiscal Year?
STDCAPCR	Standard capital credit membership	Currency / \$	Yes	Yes	Yes
FFDELTA	Delta frequent flier miles	Credit Unit / Miles	No	No	No
FREEPPV	Free pay-per-view movies	Credit Unit / Movies	Yes	No	No

The following points explain the settings for each membership above:

- The capital credit membership uses a currency of US dollars for its units. Over time, capital credits are allocated and retired. The overall balance of the credits and debits should be calculated and displayed for information purposes. It is common for the allocation and retirement of capital credits to affect the GL, and as a result, a membership SA is required for posting these financial effects. Finally, in a capital credit situation, the allocation is typically related to a specific fiscal year. When calculating and displaying balances, the balance for each fiscal year must also be available. As a result, the fiscal year must be set to required.
- For the frequent flier membership type, a separate membership type must be created for each different airline. This is because a separate membership must exist for each separate airline in order to keep track of the accumulated miles correctly. The membership type does not have a balance because the accumulated miles are interfaced to the airline and the airline keeps track of the balance. In this example, no SA is required because an assumption is being made that the creation of frequent flier miles does not affect the GL. (Accumulating miles is no liability or expense for the company. The miles are simply accumulated on behalf of a third party.) Finally, the frequent flier miles do not need to indicate a fiscal year.
- For the free pay-per-view movies membership, we assume that the credits are redeemed from within the system. For example, perhaps a calculation rule calculation algorithm or pre-processing calculation rule redeems the free pay-per-view movies over time as customers are billed for the movies. As a result, the membership should have a balance. In this example, no SA is required because an assumption is being made that any affect on the GL is posted at the time the free movies are redeemed (i.e., when calculating a bill). (This is just an example. It's possible that you may want to post to the GL when free movies are accumulated to mark a payable for the company.) Finally, free pay-per-view movies do not need to indicate a fiscal year.

More options must be considered for each membership type.

Consider Whether the Membership Should Indicate Subcategories

For some types of memberships, the amount of each service credit event is further grouped by a subcategory. Refer to [Events May Indicate a Subcategory](#) for more information.

Use subcategories if multiple subtypes of credits may be accumulated and redeemed and if balances need to be tracked for each subcategory.

For our sample membership types, only the capital credits type should indicate subcategories.

Membership Type	Description	Subcategories
STDCAPCR	Standard capital credit membership	Distribution Transportation
FFDELTA	Delta frequent flier miles	
FREEPPV	Free pay-per-view movies	

Determine the Types of Service Agreements Linked to the Membership

Memberships typically exist to reward customers for participating in standard service with the company. Refer to [How Are Service Credits Earned?](#) for more information. If memberships of this type are related to standard service for your company, determine the SA types for these service agreements.

NOTE:

Standard Service SA Types. These service agreement types are probably different from the type of service agreement you may link as the membership SA. The membership SA is used for miscellaneous charges like general ledger posting. These SA types are related to electricity service, gas service, etc.

For our three examples, assume that the company provides electric service and cable. The capital credits are related only to electric service, frequent flier miles are accumulated for a combination of services, and free pay-per-view movies are related only to cable service.

Membership Type	Description	SA Types
STDCAPCR	Standard capital credit membership	E-RES (Electric Residential) AL-RES (Area Lighting)
FFDELTA	Delta frequent flier miles	E-RES (Electric Residential) AL-RES (Area Lighting) CABLE -RES (Cable Residential)
FREEPPV	Free pay-per-view movies	CABLE-RES (Cable Residential)

Consider Special Functionality Needed When Adding, Activating or Inactivating a Membership

Should a letter or bill message be generated when a membership is created or activated? Should anything happen when a membership is inactivated? If so, you may need to define one or more algorithms to be plugged in on the membership type.

For our examples, let's say that a bill message is generated when a frequent flier or free pay-per-view membership is created. Let's say that a letter is generated when a capital credits membership is activated. We also assume that when a capital credits membership becomes inactive, any outstanding balance is redeemed. For the frequent flier miles and pay-per-view memberships, we assume nothing special occurs when the membership becomes inactive.

FASTPATH:

Refer to [Lifecycle of a Membership](#) for more information about the various status values for a membership.

Membership Type	Description	Algorithm System Event	Algorithm
STDCAPCR	Standard capital credit membership	Membership Activation	Send Letter
		Membership Inactivation	Redeem Balances
FFDELTA	Delta frequent flier miles	Membership Creation	Generate Bill Message
FREEPPV	Free pay-per-view movies	Membership Creation	Generate Bill Message

NOTE:

Sample Algorithms. The base package does not provide sample algorithms for all the above examples. Refer to [Service Credit Membership Type - Algorithm](#) for more information about the algorithms provided with the system.

Determine Whether The Membership Requires Additional Information

Is there any information about your membership that you need to capture that is not already provided by the base system logic? If the answer is yes, you may need to define characteristics for your membership. Use the [characteristic collection](#) on the membership type to define the types of characteristics allowed for memberships of this type. You may also define default values for you membership's characteristics.

Designing Your Service Credit Event Types

Now that you have designed your membership types, you need to design the types of service credit events that may be created for your membership.

In many cases, a credit event should cause additional functionality to occur. Algorithms are executed when an event is completed and when an event is canceled and are used to perform additional functionality. The following points illustrate possible algorithms that may be needed when a credit event is completed.

- Validate the event as compared to other events. For example, perhaps a new event should never cause the overall membership balance to fall below zero. You could use an algorithm on the service credit event type to check this condition.
- Create an adjustment that posts to the general ledger. Your credit event may not affect the customer's balance when created, but perhaps it should have an effect on the general ledger. For these types of credit events, you may need to create an adjustment to post to the GL.
- Create adjustments to affect the customer's balance. When "redeeming" a credit, you may need to transfer a monetary amount to one or more of the customer's service agreements.
- Create a bill message. Perhaps when a credit is accumulated, you want to inform the customer via a message on the bill indicating the credit amount. A temporary bill message is added to one of the membership's accounts.

NOTE:

One Account Message of the Same Message Type. The temporary bill message collection on the account allows only one bill message of the same bill message type. If it's possible for multiple types of events to be generated for the same account, consider creating a different bill message type for each event type.

- Stamp a batch code and batch run number onto the event. This would be used when your event information needs to be interfaced to an external system.

The following points illustrate possible algorithms that may be needed when a credit event is canceled.

- Validate the event as compared to other events. For example, perhaps canceling an event should never cause the overall membership balance to fall below zero.
- Cancel adjustments that may have been created when the event was completed.

To illustrate examples of when to use some of the algorithms above, we'll design event types for the membership types designed above.

Designing Capital Credit Event Types

The following event types illustrate typical events for a capital credits membership.

SC Event Type	Description	Membership Type	Algorithm System Event	Algorithm
ALLOCATECCR	Capital credit allocation	STDCAPCR	Event Creation	Create Simple Adjustment
			Event Creation	Generate Bill Message
			Event Cancellation	Validate Balance Not < Zero
			Event Cancellation	Cancel Related Adjustments
RETIRECCR	Retire capital credit (apply to customer's balance)	STDCAPCR	Event Creation	Validate Balance Not < Zero
			Event Creation	Create Adjustments to Affect Customer's Balance
			Event Cancellation	Cancel Related Adjustments
FORFEITCCR	Forfeit capital credit (retirement not applied to customer's balance)	STDCAPCR	Event Creation	Validate Balance Not < Zero
			Event Creation	Create Simple Adjustment (affect GL only)
			Event Cancellation	Validate Balance Not < Zero
			Event Cancellation	Cancel Related Adjustments

These event types assume the following:

- When credit events are allocated, the customer is notified via a bill message, the general ledger is affected so a simple GL only adjustment is created. If an event of this type is canceled, any adjustments that were created should be canceled. The event amount should always be positive, so checking that the membership balance does not fall below zero is only checked for event cancellation.
- When capital credits are retired, it's possible that the full membership balance is not applied to the customer's balance. For the portion of the retirement that does affect the customer's balance, you need an algorithm that applies the credits to the customer's service agreements via adjustments. Cancellation of this event should cause any related adjustments to be canceled. It's assumed that the amount of this event is a credit so checking that the membership balance does not fall below zero is only checked for event completion.
- For the portion of the retirement that is not applied to the customer's balance, the event amount should simply affect the GL so a GL only adjustment is created. Cancellation of this event should cause any related adjustments to be canceled.

It's assumed that the amount of this event is a credit so checking that the membership balance does not fall below zero is only checked for event completion.

NOTE:

Service credit event types are independent of subcategories. A capital credits membership typically uses subcategories. When events are created for different subcategories, the same service credit event type may be used. As a result, all subcategories use the same event completion and event cancellation algorithms.

Designing Frequent Flier Event Types

The following event types illustrate typical events for a frequent flier membership:

SC Event Type	Description	Membership Type	Algorithm System Event	Algorithm
ADDMILES	Add miles to the membership	FFDELTA	Event Creation	Generate Bill Message
			Event Creation	Populate Batch Information
			Event Cancellation	Event may not be canceled.

This event type assume the following:

- A new event should generate a bill message.
- Information about the event amount should be interfaced to an external system so batch information should be populated when the event is completed.
- Events of this type may not be canceled because the information is interfaced to an external system. Rather, to reverse an event, simply create a new event whose amount is a credit. This credit amount is also interfaced to an external system.

NOTE:

Sample Algorithm. The system does not provide a sample cancellation algorithm that prevents the event from being canceled.

- There is no validation to ensure that the balance does not fall below zero. Recall that this membership was defined as not requiring a balance.

Notice that only one type of event has been defined for this membership. That is because the credits for this membership are not redeemed via this system. Rather they are accumulated on behalf of an external system.

Designing Pay-per-view Event Types

The following event types illustrate typical events for a free pay-per-view movies membership:

SC Event Type	Description	Membership Type	Algorithm System Event	Algorithm
ADDPV	Free pay-per-view movies	FREEPPV	Event Creation	Generate Bill Message
			Event Cancellation	Validate Balance Not < Zero
REDEEMPPV	Redeem pay-per-view movies	FREEPPV	Event Creation	Validate Balance Not < Zero

These event types assume the following:

- Any type of new event should generate a bill message.
- For adding free movies, it is assumed that the quantity is positive so when the event is cancelled, the algorithm verifies that the balance does not become negative.
- For redeeming free movies, it is assumed that the quantity is negative so when the event is completed, the algorithm verifies that the balance does not become negative.
- These events do not affect the general ledger and they do not directly affect the customer's balance so no adjustment algorithm is needed.

How Are Service Credit Events Created?

Now that you have designed the behavior of your service credit events, an important issue is to determine how these events are created. Consider your business practice for each type of membership.

Let's use our sample memberships to work through different ways that you may create service credits.

Service Credits for Capital Credit Memberships

For a capital credit membership, capital credit allocations are calculated once a year based on billing history and the cost of service. To accomplish this, a background process calculates the amount and creates the service credits. Refer to [allocating capital credits](#) for a sample process provided with the base package.

Credits are redeemed via the retirement process. The company determines when to retire capital credits based on analysis of their financial situation. This retirement process is also handled by a background process. Refer to [capital credit retirement](#) for a process provided with the base package.

For capital credits memberships, special functionality is required when a member dies. The capital credits are considered part of the person's estate and may need to be retired and applied to a beneficiary's account. This process depends on the company's business practice. However, typically, the membership status would change to inactive so that new capital credit allocations are not created for the membership. The system provides a sample inactivation algorithm called [SCMI-RB](#) that transfers part or all of the outstanding credit balance to the member's service agreements. From there, a user can cut a check to the beneficiary.

NOTE:

Some Credits Are Never Retired. For many cooperatives, some types of allocated credits are never retired. Refer to [Partial Retirement](#) for more information.

Service Credits for Frequent Flier Memberships

In our example, frequent flier miles were related to both the electricity and phone service. Let's assume that frequent flier miles are accumulated every \$x spent on electricity and phone service. In this example, a bill completion algorithm creates service credits based on the bill segment amounts for these service agreements. Refer to [Service Credits Earned Through Billing](#) for more information.

As mentioned before, the frequent flier miles are not redeemed using this system, but are interfaced to a third party for redemption.

Service Credits for Free Movies Memberships

NOTE:

Sample Algorithms. No base package algorithm types are provided to support the logic described in this example.

In this scenario, let's suppose that a customer receives three free pay-per-view movies when signing up for new service. To handle this, perhaps an SA creation algorithm creates the membership and the service credits when the service is started. Or perhaps you want to wait until the first bill is generated and a bill completion algorithm is used to generate the first credit. It depends on your business practice.

For redeeming the free pay-per-view movies, it is assumed that the movies are credited during billing after it is determined that the customer has been billed for a movie. The number of movies used for the membership is reduced until all the free movies are used.

FASTPATH:

Refer to [Service Credits Redeemed Through Billing](#) for more information.

Setting Up Service Credit Options

Setting Up Credit Units

Credit unit is used for service credit membership types whose [events](#) record non-monetary units. Open **Admin > Credit Unit** to set up credit units.

Description of Page

The following fields display for each credit unit:

Credit Unit The unique identifier of the credit unit.

Description The description of the unit. This also acts as a label for the unit when displaying information about a service credit event.

Symbol / Label Position Indicates whether or not the label for this credit unit appears as a Prefix or as a Suffix to the service credit event amount.

Decimal Positions Indicates the number of decimal positions used for this credit unit. This information should be used by any algorithm or background process that creates a service credit event to determine how to store the event amount. It is also used to correctly display the service credit amounts.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CR_UNIT](#).

Setting Up SC Membership Inactive Reasons

The service credit membership inactive reason must be specified when the status of a service credit membership changes to inactive. Open **Admin > Service Credit > Membership Inactive Reason > Add** to set up service credit inactive reasons.

Description of Page

The following fields display for each inactive reason:

Inactive Reason The unique identifier of the service credit membership inactive reason.

Description The description of the inactive reason.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SCM_NCTV_RSN](#).

Setting Up Service Credit Membership Types

SC Membership Type - Main

Service credit memberships have a membership type. Open **Admin > Service Credit > Service Credit Membership Type > Add** to define the membership type.

Description of Page

Enter a unique **Service Credit Membership Type** and **Description** for each membership type.

Use the **SA Requirement** flag to indicate whether a [miscellaneous SA](#) must be linked to memberships of this type. The possible values are SA Required and SA Not Allowed .

NOTE:

The value defaults to SA Required.

Use the **Fiscal Year Requirement** flag to indicate whether events linked to memberships of this type should indicate a [fiscal year](#). The possible values are Fiscal Year Required and Fiscal Year Not Allowed.

NOTE:

Default Note. The value defaults to Fiscal Year Not Allowed.

Use the **SCM Event Balance** flag to indicate whether or not a [balance of service credit events](#) linked to memberships of this type should be calculated and displayed. The possible values are Has a Balance and No Balance.

NOTE:

Default Note. The value defaults to Has a Balance.

Use the **SC Membership Type Unit** flag to indicate whether or not the unit for the event amounts for events linked to memberships of this type is Currency or Credit Unit. When the unit is currency, indicate the **Currency Code**. When the unit is credit unit, indicate the **Credit Unit**.

If events linked to memberships of this type must indicate a subcategory, enter a valid **Subcategory Name** and **Description** for each subcategory applicable to this membership.

FASTPATH:

Refer to [Events May Indicate a Subcategory](#) for more information.

Use the **SA Types** grid to indicate the types of [service agreements that contribute to memberships](#) of this type. Indicate the **CIS Division** and **SA Type** for each type of service agreement that is related to a membership of this type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_SCM_TYPE](#).

SC Membership Type - Algorithm

Open **Admin > Service Credit > Service Credit Membership Type > Search** and navigate to the **Algorithm** page to define any algorithms that are associated with a membership type.

Description of Page

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (descriptions of all possible events are provided below).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** for which you can define algorithms.

System Event	Optional / Required	Description
Membership Creation	Optional	Use this system event for algorithms that are executed when a membership is created. Click here to see the algorithm types available for this system event.
Membership Activation	Optional	Use this system event for algorithms that are executed when a membership becomes active. Click here to see the algorithm types available for this system event. NOTE: Creating Memberships In Active Status. A membership may be created in either pending status or in active status, based on your business practice. For memberships created in active status, the system executes the membership creation algorithms and the membership activation algorithms.
Membership Inactivation	Optional	Use a system event of Membership Inactivation for algorithms that are executed when a membership becomes inactive . Click here to see the algorithm types available for this system event.

SC Membership Type - Characteristics

To define characteristics that can be defined for service credit memberships of a given type, open **Admin > Service Credit > Service Credit Membership Type > Search** and navigate to the **Characteristics** page.

Description of Page

Use the characteristics collection to define characteristics that can be defined for service credit memberships of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on service credit memberships of a given type. Turn on the **Default** switch to default the **Characteristic Type** when service credit memberships of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

Setting Up Service Credit Event Types

Service credit events created for a service credit membership have an event type. Open **Admin > Service Credit Event Type > Add** to set up service credit event types.

Description of Page

Enter a unique **Service Credit Event Type** and **Description** for each event type.

Indicate the **Service Credit Membership Type** to which this event type belongs.

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (descriptions of all possible events are provided below).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** for which you can define algorithms.

System Event	Optional / Required	Description
Event Completion	Optional	Use a system event of Event Completion for algorithms that are executed when an event is completed. Refer to An Event May Cause Other Actions to Occur for more information. Click here to see the algorithm types available for this system event.
Event Cancellation	Optional	Use this system event for algorithms that are executed when an event is canceled. Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_SC_EVT_TYPE](#).

Service Credit Examples

In this section, we provide examples of how to define your control tables to support functionality related to different types of service credit memberships. While your company may not define your environment exactly the same way, this section should help solidify your understanding of how to set up your company's service credit options.

Defining Control Tables for a Refundable Fee

If your membership requires payment of a [fee](#) that is refundable, you must define an SA type to use for the refundable fee.

Adjustment Type for Refundable Fee

This example assumes that the fee is set up similarly to a deposit. The adjustment used to charge the fee would affect the current balance, but not the payoff balance or the general ledger because the fee is not considered a "receivable", rather it is an amount collected and held for the customer.

Create an [adjustment type](#) for levying the fee. Indicate the fee amount and indicate the adjustment FT creation algorithm [ADJT-CA](#), which affects the current balance, but not the payoff balance or the general ledger.

NOTE:

Adjustment Type Profiles. Be sure to add this adjustment type to an appropriate [adjustment type profile](#) and ensure this profile is linked to your SA type.

SA Type for Refundable Fee

Create an SA type to use for the fee. This SA type should be marked as not billable.

Indicate the appropriate adjustment type profile that includes the adjustment type to levy the fee.

Indicate an appropriate payment segment type that references the [PSEG-NM](#) payment segment FT algorithm. This algorithm affects the payoff amount and current amount by the payment amount, which should cause the current amount to become zero and the payoff amount to become a credit for the fee amount when the fee is paid.

Start Option for Refundable Fee

For this SA type, define a start option that causes an adjustment to be levied as part of the start service process.

FASTPATH:

Refer to [Including The Membership Fee](#) on campaigns and packages for more information about levying the fee via the order transaction.

Algorithm for Refunding the Fee

The base product provides an algorithm type [SAST-RF](#) that refunds a fee when [service agreements that contribute to the membership](#) are stopped. You must create an algorithm for this algorithm type and enter the SA type created above as an input parameter.

This SA stop algorithm must be plugged in on all SA types that you defined for the [membership type](#).

Defining Control Tables for a Nonrefundable Fee

If your membership requires payment of a [fee](#) that is not refundable, you can set this up in two ways:

- You can create a special SA type to handle charging the fee. For this SA type, define a start option that causes an adjustment to be levied as part of the start service process. This adjustment contains your fee. You may use this option if the membership is related to multiple types of services and not all services need to be present in order to create the membership.
- You can levy an adjustment on one of the other service agreements that is being started. To do this, you would use a start option to define an adjustment to be levied as part of the start service process. You may use this option if the membership is related to a single service (for example, free pay-per-view movies).

For our example, we will set up the SA for the nonrefundable fee with a separate 'fee' SA type.

Adjustment Type for Nonrefundable Fee

Create an [adjustment type](#) for levying the fee. Indicate the fee amount and indicate the adjustment FT creation algorithm [ADJT-NM](#), which affects the current balance and payoff balance by the adjustment amount, and affects the general ledger.

NOTE:

Adjustment Type Profiles. Be sure to add this adjustment type to an appropriate [adjustment type profile](#) and ensure this profile is linked to your SA type.

SA Type for Nonrefundable Fee

Create an SA type to use for the fee. This SA type should be marked as not billable.

Indicate the appropriate adjustment type profile that includes the adjustment type to levy the fee.

Indicate an appropriate payment segment type that references the [PSEG-NM](#) payment segment FT algorithm. This algorithm affects the payoff amount and current amount by the payment amount, which causes the current amount to become zero and the payoff amount to become a credit for the fee amount when the fee is paid.

Start Option for Nonrefundable Fee

For this SA type, define a start option that causes an adjustment to be levied as part of the start service process.

FASTPATH:

Refer to [Including The Membership Fee](#) on campaigns and packages for more information about levying the fee via the order transaction.

Defining an SA Type for Miscellaneous Transactions

Does your membership require a [service agreement](#) to support miscellaneous transactions? If so, you need to consider the SA type to use for this service agreement. This SA type should be marked as not billable.

If you choose to use the order transaction to set up the membership, this SA type must be defined on the [algorithm that creates the membership](#).

It is possible that you may require a start option for this SA type, for example if you want to define a characteristic for service agreements of this type.

Using Campaigns/Packages to Set Up Membership

If enrollment in a membership is a common occurrence for your customers when starting service, you should consider using the [order](#) page to start service for the customer and to create the membership as well.

NOTE:

This section only makes sense if you are familiar with the [Sales and Marketing](#) chapter.

The recommendation is to use a question/miscellaneous field to ask the customer service representative to indicate the appropriate membership type. Algorithms validate this membership type and use it to create a membership of that type.

This section walks you through how to set up the campaign / package required to support this.

Column Reference for Membership Type

In order to ask the customer service representative to define an appropriate service credit membership type, you must define a column reference for the membership type.

Add a new column reference with the following information:

- Column Reference: SCM-TYPE
- Description: SC Membership Type
- FK Reference: SCM TYPE
- Long Description: Service Credit membership type to use when creating a service credit membership at start time.

NOTE:

Column Reference Algorithms. This column reference indicates a validation algorithm and a posting algorithm. However, we have not defined them yet so for now simply save this information.

Algorithms to Create Membership via Order

The base product provides two algorithm types to support the creation of a membership record via the order page: a validation algorithm type and a posting algorithm type.

Validate Membership Information

This algorithm is a column reference validation algorithm that checks that an input membership type is valid. Refer to [CRVL-ME](#) for more information. You must define an appropriate algorithm for this algorithm type, and on that algorithm you must define the column reference used to identify the membership type. For example:

- Algorithm: VAL MEM TYPE
- Description: Validate SC Membership Type
- Algorithm Type: CRVL-ME
- Parameter1: (Column Reference for Membership Type): SCM-TYPE

Post Membership Information

This algorithm is a column reference posting algorithm that creates a membership using the membership type indicated by the user. Refer to [CRPS-ME](#) for more information. You must define an appropriate algorithm for this algorithm type, and on that algorithm you must define the column reference used to identify the membership type. For example:

- Algorithm: CREATE MEMBRSH
- Description: Validate SC Membership Type
- Algorithm Type: CRPS-ME
- Parameter1: (Column Reference for Membership Type): SCM-TYPE
- Parameter2: (Service Credit Membership Status): 10 -Pending or 20 -Active. Refer to [Lifecycle of a Membership](#) for more information.
- Parameters3-5: (Division, SA Type, Start Option): Indicate the information needed to create an SA for miscellaneous transactions.

NOTE:

This algorithm first looks for an existing service agreement of this division / SA type linked to the membership's accounts. If one exists, it uses that SA to link to the membership. If an SA does not exist, it creates one with the input division, SA and (optional) start option.

Update the Column Reference. Now that you have defined the validation and posting algorithms, return to your [column reference for membership type](#) and define the algorithms there.

Although your company may support multiple types of memberships, this column reference and its algorithms have been designed such that only one column reference for membership type would be needed to set up any type of membership. Although the posting algorithm for the membership type column reference indicates an SA type to use for miscellaneous transactions, a service agreement is only created and linked if your membership type indicates that an SA is required. As a result, you may use the same column reference for both memberships that require an SA and those that don't require an SA. However, if you have different membership types that require an SA and each uses a different SA type or a different start option for the membership SA, you need to define more than one posting algorithm and, as a result, more than one column reference.

Define a Campaign for Creating a Membership When Starting Service

Many factors must be considered when [designing your campaigns and packages](#). The possible creation of a membership when using the order page is simply another factor to consider.

If you plan to define a question/miscellaneous field to capture a service credit membership type, the available packages linked to the campaign should be ones that are related to memberships. For example, if your membership is related to electric service, it doesn't make sense to create a membership for a campaign designed to generate a one-time charge.

When defining a question/miscellaneous field, you must indicate its applicability. Consider whether a membership type is required, optional or only applicable for certain packages. This helps ensure that your users capture this information when appropriate.

Including the Membership Fee

You must determine the best way to setup your campaign/package to handle the levying of your membership fee, if applicable. Consider some of the following questions.

- If the membership has a fee, is it a refundable fee or a non-refundable fee?

- Is the fee always applicable for the membership? Or is it waived under certain conditions? For example, maybe the fee is applicable if a customer signs up for a single service, but the fee is waived if the customer signs up for two or more services.

The SCM type column reference and question/miscellaneous field are set up to ask the user what type of membership to create. Because the applicability of the fee may differ for each membership type, you should carefully consider the campaign / package setup to levy the fee correctly.

- If a fee is always applicable for a membership, you may consider creating a membership creation algorithm that creates a fee SA with a start option to levy the fee when the membership is created.

NOTE:

Sample Algorithm. The base product does not provide a membership creation algorithm to do this.

- If the fee is not always applicable for a membership, you must determine when the fee is applicable. The recommendation is to include the fee SA type in the SAs-to-create collection for the appropriate package. For example,
 - If the fee is applicable when the customer signs up for a single service, you should define a package for each single service that includes one SA to create for the single service and the fee SA as another SA to create.
 - If the fee is waived when the customer signs up for two or more services, you should define a package for each combination of the multiple services. These packages do not include the fee SA in the SAs to create.
- If the customer signing up for service is a former customer who has returned, perhaps the fee was paid earlier when the customer originally had service. In this case, maybe your business practice is to waive the fee at this time. To do this, you should set up a question/miscellaneous field for the user to mark that this is a returning customer. Based on the answer to this question, perhaps only packages that do not levy any fee are eligible for selection.

Defining Another Person for Your Account

It is common for a capital credit membership to define more than one person for the account being turned on and linked to a membership. A typical example is a married couple. Both spouses are indicated on the account and financially responsible persons and as a result, both are considered [members](#).

Column reference algorithms have been provided by the base product to enable linking a second person to your account via the [order](#) page.

This section walks you through how to set up the column reference and campaign to support this.

NOTE:

Linking A Second Person to the Account. This logic is not restricted to service credit functionality. Any campaign may be designed to include the ability to link a second person to the account being started.

Column References for Additional Person

In order to ask the customer service representative to link an additional person for the account, you must define several column references to use as questions/miscellaneous fields.

- We should assume that the person may already exist in the database. As a result, a question to record the person ID is needed.
- If the person does not already exist, the user should capture the person's name, an ID type and an ID number. Questions for these three fields are needed. The system uses this information to create a new person.

- Whether we are using an existing person or creating a new one, the person's link to the account must include an account relationship type. A question to record the account relationship type is needed.

Add a new column reference for person ID:

- Column Reference: PERSON-ID
- Description: Person ID
- FK Reference: PER
- Long Description: Person ID of the additional person to link to the order's account.

Add a new column reference for person name:

- Column Reference: PERSON-NAME
- Description: Person Name
- FK Reference: not applicable
- Long Description: Name of a new person to create.

Add a new column reference for identifier type:

- Column Reference: PER-ID-TYPE
- Description: Identifier Type
- FK Reference: ID TYPE
- Long Description: Identifier type for the primary ID of the additional person to link to the order's account.

Add a new column reference for identifier number:

- Column Reference: PER-ID-NUM
- Description: Person ID Number
- FK Reference: not applicable
- Long Description: Identifier number for the primary ID of the additional person to link to the order's account.

Add a new column reference for account relationship type:

- Column Reference: ACCT-REL-TYPE
- Description: Account Relationship Type
- FK Reference: ACCT REL
- Long Description: Relationship type to use for the link between the additional person and the order's account.

NOTE:

Column Reference Algorithms. One of the column references above must indicate a validation algorithm and a posting algorithm. However, we have not defined them yet so for now simply save this information. We recommend using the account relationship type record because that is used for all additional persons.

Algorithms to Link Additional Person via Order

The base product has provided two algorithm types to support the linking of an additional person to the account via the order page: a validation algorithm type and a posting algorithm type.

Validate Addition Person Information

This algorithm is a column reference validation algorithm which checks that either a person id or person name, a valid ID type and ID number are provided and that a valid account relationship type is provided. Refer to [CRVL-PE](#) for more

information. You must define an appropriate algorithm for this algorithm type and on that algorithm you must define the column reference used for the five fields required for this validation. For example:

- Algorithm: VAL ADD PER
- Description: Validate Additional Person Info
- Algorithm Type: CRVL-PE
- Parameter1: (Column Reference for Person ID): PERSON-ID
- Parameter2: (Column Reference for Person Name): PERSON-NAME
- Parameter3: (Column Reference for Person ID Type): PER-ID-TYPE
- Parameter4: (Column Reference for Person ID Number): PER-ID-NUM
- Parameter5: (Column Reference for Account Relationship Type): ACCT-REL-TYPE

Post Addition Person

This algorithm is a column reference posting algorithm that may link an existing person to the order's account or create a new person and link that person to the order's account. Refer to [CRPS-PE](#) for more information. You must define an appropriate algorithm for this algorithm type, and on that algorithm you must define the column reference used for the five fields required for this logic. For example:

- Algorithm: LINK ADDNTL PER
- Description: Link additional person to order's account
- Algorithm Type: CRPS-PE
- Parameter1: (Column Reference for Person ID): PERSON-ID
- Parameter2: (Column Reference for Person Name): PERSON-NAME
- Parameter3: (Column Reference for Person ID Type): PER-ID-TYPE
- Parameter4: (Column Reference for Person ID Number): PER-ID-NUM
- Parameter5: (Column Reference for Account Relationship Type): ACCT-REL-TYPE

NOTE:

Update the Column Reference. Now that you have defined the validation and posting algorithms, return to your [column reference for account relationship type](#) and define the algorithms there.

Design a Campaign to Include Linking an Additional Person

Any campaign related to a specific account may include the questions and miscellaneous fields defined here to create/link an additional person to the account.

Simply create entries in the questions and miscellaneous fields collection to prompt the user to ask for the required information. A question should exist for each column reference created above. The following table illustrates a possible setup.

Seq	Description	Prompt on Order	Column Reference	Dependency
10	Account relationship type for additional person.	If you would like to link another person to this account, please enter an account relationship type.	ACCT-REL-TYPE	Must have account
20	Person ID for additional person	If the additional person already exists in the	PERSON-ID	Must have account

		database, enter the person id.		
30	Person name for additional person	If you would like to create a new person, please enter the person name.	PERSON-NAME	Must have account
40	ID type for additional person	Please enter the ID type of the new person.	PER-ID-TYPE	Must have account
50	ID Number for additional person	Please enter the ID number of the additional person.	PER-ID-NUM	Must have account

Service Credits Earned When Starting Service

For some memberships, you may want to add service credits when starting the program. For example, the customer gets three free pay-per-view movies when signing on for cable service. Or a customer receives 500 frequent flier miles for signing up for a combination of gas and electric service.

There are various ways that you can accomplish this. You should work with your implementers to consider the various options to determine the method that best suits your business practice.

- You could use an SA activation algorithm to create a membership and an event for the initial points. Use this method if the points are related to a single type of service agreement and the points are earned automatically when starting service (i.e., without any human decision to be made).
- You could use a membership creation or membership activation algorithm to generate a service credit event automatically. This method assumes that the decision to create the membership has been made and that the free initial points are always earned for this type of membership (i.e., regardless of the type of service created).
- You could use questions and miscellaneous fields for a campaign/package to determine a customer's eligibility for participation in the membership and for the initial free points. A column reference posting algorithm could create the membership and/or the service credit event for the free points based on the answers to the questions.
- Perhaps the initial free points are only earned after the first bill is generated. Use a bill completion algorithm to generate the initial points.

You may think of other plug-in spots that could be used to generate free initial points based on your business needs.

Service Credits Earned Through Billing

For some memberships, you may accumulate points as a result of billed amounts for other services. For example, perhaps your customers earn one frequent flier mile for every \$10 spent on the combined electricity, gas and water bills each month.

To accomplish this, you must design an algorithm to be executed at billing time. There are various plug-in spots executed at billing time that you may use, but the recommended plug-in is a post completion algorithm on the customer class. This plug-in is executed after all bill segments are frozen and most of the completion logic has occurred.

Your algorithm should determine the [service agreements that contribute to the membership](#) and calculate the service credit amount for those service agreements bill segments.

This algorithm should also consider what to do when bill segments that contributed to the event are canceled. The algorithm provide with the base product [CBCM-SC](#) checks to see if any canceled bill segments are referenced on any previous events. If so, it includes the canceled amount on the new event. This may cause the new event to be a negative amount.

The assumption is that over time, earned credits will compensate for the negative event amount. For a membership that [interfaces information to a third party](#), it is assumed that the negative event amount is also interfaced.

Service Credits Redeemed Through Billing

For some memberships, your customer may redeem their earned points by receiving a discount on their bill. For example, if your cable customer has earned one free pay-per-view movie, you can give them a credit the next time they get billed for a pay-per-view movie.

The base product has not provided any algorithms to credit a bill based on earned service credits. This section will identify considerations your implementers should follow when designing algorithms to redeem service credits through billing.

The recommendation is to credit the customer's bill by generating an adjustment using a pre-bill completion algorithm. This algorithm's responsibilities are as follows:

- Determine if the appropriate bill segment(s) contain the charges that need to be credited. For example, if the service credit is for a free pay-per-view movie, determine if the customer has been billed for a pay-per-view movie.
- Determine if the customer's current service credit balance for this program. For example, how many free pay-per-view movie credits are left?
- Create an adjustment to credit the appropriate service agreement by the eligible credit amount.
- Update the service credit membership balance by creating a new SC event with a credit for the number of points redeemed. Link the adjustment to the event as a contributed to FT.
- Consider cancel/rebill situations. If a cancel/rebill has occurred, determine if there is a change to the redeemed credits. For example, if the original bill had one pay-per-view movie that was credited and the new bill also has one pay-per-view movie, no change is needed to the service credit balance. If the original bill had more pay-per-view movies than the new bill, perhaps one or more redeemed service credit points should be reinstated. If the original bill had fewer pay-per-view movies than the new bill, perhaps more points should be redeemed.

You may wonder why we don't recommend crediting the customer's bill while generating the bill segment. For example, use pre-processing calculation rule to determine if any points should be redeemed and use a calculation rule to generate a bill calculation line with the credit amount.

The reason for this is that cancel/rebill logic is not straightforward. Algorithms executed during rate application should NOT perform any updates, such as updating the service credit membership balance. The balance should be updated using a bill segment freeze algorithm or a bill completion algorithm.

When a cancellation occurs, the service credit balance should be updated to reinstate the redeemed points. Again, this should occur when the cancellation is frozen or at bill completion time. If you perform a cancel/rebill, the calculation of the rebill segment does not have the up-to-date information about the service credit balance because the reinstatement of the points by the canceled segment has not occurred yet.

Designing Your Rate Options for Capital Credits

The capital credit allocation background process relies on certain data configuration in order to function correctly. This section identifies the required data setup.

FASTPATH:

Refer to [Allocating Capital Credits](#) for more information.

Identifying SQ and Sales Information for Historical Bill Segments

To allocate capital credits, the background process retrieves billing history for each SA that contributes to the membership for the given fiscal year. The process needs to calculate the service quantity (SQ) amount billed and the sales amount billed for the SA in that year.

NOTE:

Sales Amount. The sales amount refers to the monetary amount billed. For a capital credit allocation, this amount would generally exclude taxes and may exclude other line item amounts from the bill.

In order to calculate the amounts correctly, the background process must determine which bill calculation lines for each bill segment contain the SQ and/or sales information. Characteristics on the bill calculation lines identify which bill lines should be used.

The rest of this section uses examples to illustrate how you may configure your rate options to support this.

Define Characteristics for SQ and Sales

Char Type for identifying bill lines that hold **SQ information**

- Char Type: CCA-SQ
- Description: Capital credit allocation usage
- Type of Char: Pre-defined
- Values: Y
- Char Entities: Calculation Rule, Bill Calculation Line

Char Type for identifying bill lines that hold **sales information**

- Char Type: CCA-SALES
- Description: Capital credit allocation sales
- Type of Char: Pre-defined
- Values: Y
- Char Entities: Calculation Rule, Bill Calculation Line

Define Calculation Rules

Identify each rate schedule that may be linked to a service agreement that contributes to a capital credit membership.

For each of these rate schedules, identify the calculation rules whose resulting bill calculation line will contain a billable service quantity that should be included in the SQ calculation for allocating capital credits. For each calculation rule that qualifies, define the CCA-SQ char type (defined above) and a char value of Y.

For each rate schedule, identify the calculation rules whose resulting bill calculation line amount should be included in the sales calculation for allocating capital credits. For each calculation rule that qualifies, define the CCA-SALES char type (defined above) and a char value of Y.

NOTE:

Characteristic Information. The system automatically copies characteristic info from a calculation rule to its resulting bill calculation line if the char type entities include both calculation rule and bill calc line.

Define Batch Control Parameters

The background process to allocate capital credits [CPCRALOC](#) receives the char type and char value to identify the bill calculation lines that contain the SQ and sales amounts. Once you have your characteristics defined, update your batch control to include these values as default parameter values.

Designing Bill Factors for Credit Allocation

The [capital credit allocation](#) process uses an allocation factor in its calculation. A typical capital credit membership may define multiple subcategories, meaning that allocation amounts are calculated each year for the multiple subcategories. The calculation is the same for each subcategory, but the allocation factor differs.

The process has been designed to calculate the allocation for a single subcategory. If your organization requires allocations calculated for multiple subcategories, the process must be run for each subcategory. The allocation process receives a bill factor as an input parameter. As a result, a different bill factor should be set up to define the allocation factor for each subcategory.

For each subcategory, the allocation factor may differ further for the type of customer. For example, the allocation for a commercial customer may differ from the allocation factor for a residential customer. The allocation background process expects the bill factor for each subcategory to define a characteristic type of revenue class. The process determines each service agreement's revenue class by looking at the value defined on its SA type.

Following is an example of bill factors set up for a capital credit membership with two subcategories: transportation and generation.

Characteristic Type for Allocation Bill Factor

Char Type: REV-CLASS

Description: Revenue Class

Type of Char: Pre-defined

Values: (define all the valid revenue class values)

Char Entity: Bill Factor

Bill Factor for Transportation

Bill Factor Id: CCAF-TRANS

Description: Transportation Allocation Factor

Char Type: REV-CLASS

Char Source: Characteristic Collection

Char Values: (for each year, the new transportation allocation factor for each revenue class must be defined)

Bill Factor for Generation

Bill Factor Id: CCAF-GEN

Description: Generation Allocation Factor

Char Type: REV-CLASS

Char Source: Characteristic Collection

Char Values: (for each year, the new generation allocation factor for each revenue class must be defined)

NOTE:

The characteristic source is characteristic collection. It is the responsibility of the background process to determine the SA's revenue class and to pass this value into the bill factor routines to retrieve the correct bill factor value.

Estimating Allocation Factors. Often the company needs to estimate the allocation factors for the new fiscal year and may adjust the values several times until the calculated allocation amounts are satisfactory. Refer to [Allocating Capital Credit](#) for more information.

Partial Retirement

In the cooperative business, it is common to never retire certain capital credit allocation amounts. The amounts that do not retire should be assigned their own subcategory.

When executing the retirement background process, the subcategory to retire may be input to the process. If you have certain subcategories that you do not retire, you would simply run the background process for the subcategories that do retire.

Cooperatives typically retire amounts and transfer the amounts to a beneficiary when a member dies. This is known as "estate retirement". Refer to [Service Credits for Capital Credit Memberships](#) for more information. If your business practice designates that certain subcategories of allocated amounts do not get retired, this probably holds true for estate retirement as well. If that is the case, your membership inactivation algorithm should be designed to only retire the appropriate amounts by subcategory.

Interface Membership Information to a Third Party

For some memberships, you may accumulate points for a third party, for example accumulating frequent flier miles for an airline. For these types of memberships, you must interface the event information to the third party.

To interface information to a third party, you may choose one of the following options:

- Design an extract program to interface the information
- Use workflow and notification to interface the information via the XAI tool

Interface Via an Extract Program

The service credit event may indicate a batch code and batch run number. Design a program to extract event information to a third party. This extract program would select service credit events marked with its batch code and the current run number.

Your service credit event must define a completion algorithm that stamps the appropriate batch code and run number. The base product provides an algorithm type to perform this logic. Refer to [SCEC-BT](#) for more information.

Use Workflow & Notification to Interface Info

Workflow and notification allows you to send information to a third party via an extract program or via XAI. The service credit event has its own logic for interfacing via an extract program, so you would use workflow and notification only if you need to interface to the third party via XAI.

FASTPATH:

Refer to [Designing Notification Downloads](#) for more information.

Other Considerations For Interfacing Info to a Third Party

Because event information is extracted to a third party, you must consider how to handle adjustments to the event amount. For example, if your event is generated based on the creation of a bill segment, what should happen if that bill segment is cancelled?

You may want to prevent these types of events from getting canceled. Validation like this may be added via a user exit or using an event cancellation algorithm.

NOTE:

Sample Algorithms. The product does not provide any base algorithms to prevent an event from being canceled.

You should allow negative event amounts to be created so that this information may also be sent to the third party's system.

Defining Loan Options

The topics in this section describe how to set up the system to enable loan functionality.

NOTE:

Loans are optional. The system configuration requirements described in this section are only relevant if your organization loans money to customers.

The Terms Of A Loan Are Stored On A Service Agreement

Loans are initiated by creating a loan service agreement for a customer. The loan service agreement (and its SA Type) contains the loan's terms:

- The **loan amount** is held in the service agreement's Total Amount to Bill.
- The **customer's periodic payment amount** is held in the service agreement's Recurring Charge Amount.
- The **number of amortization periods** (e.g., 36 months, 240 months, etc.) is held in the service agreement's Number of Payment Periods.
- If the **interest rate** is the same for all customers with this type of loan service agreement, the interest rate is defined on the service agreement's SA type (using a bill factor). A specific interest rate can be defaulted from a start option contract

value. If a specific interest rate applies to the customer, the SA type's interest rate can be overridden by specifying a bill factor value on the customer's service agreement (where the bill factor value contains the specific interest rate for the customer).

- The SA type controls the **periodicity of the bills** (e.g., monthly or bi-weekly).

Because a loan is defined using a service agreement, the typical functionality that is controlled by the service agreement's SA type is supported, including:

- How and when it is billed.
- How payments are booked in the GL (and the payment priority relative to other service agreements).
- How its debt is monitored by credit and collections.
- How late payment charges are calculated.

Loan service agreements are created using [start / stop](#) just like all other service agreements. The start/stop transaction has special loan functionality that allows an operator to specify the service agreement-specific loan terms described above. A start option can be specified to override the SA type's interest bill factor.

NOTE:

Automatic calculation of periodic payment amount / number of periods. The system calculates a loan's periodic payment amount or number of payments (whichever is left blank). You can have the system do this on [Start/Stop Maintenance](#) (using the **Calculate** button that appears on the [start service confirmation window](#)), and on [Loan - Main](#) (by clicking the **Calculate** button). Regardless of where you do this, the calculation is performed by an algorithm on the loan's SA type. Refer to the [LPDA-SI](#) algorithm type for more information about the base package algorithm.

Payoff Balance and Current Balance for Loans

As described under [Current Amount versus Payoff Amount](#), a loan service agreement's current balance and payoff balance differ during the lifetime of the loan. Current balance contains how much the customer owes to remain current (typically their periodic payment amount), and payoff balance contains the amount the customer would have to pay to payoff the loan (typically the principal balance plus any accrued interest charges).

Unlike other SAs, loans have two accounts receivable distribution codes: long term and short term. These two codes allow the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). The current balance for a loan is always the amount of the short-term receivables. The payoff balance for a loan is always the net of the short-term receivables and the long-term receivables.

If the SA has a [special role](#) of Loan, the financial transaction algorithms supplied with the base package transfer the current amount between the long-term receivables and the short-term receivables in the GL. For example, when a bill segment is generated for the loan SA, the amount of the periodic payment is transferred from the long-term receivables to the short-term receivables. Don't worry, the examples in the following sections show exactly what these transactions look like.

An operator can see the how the bills, payments and adjustments have affected the GL, current balance and payoff balance using [SA Financial History](#).

The following sections provide examples of how adjustments, bills and payments are recorded in the GL and the subsequent effect on the current and payoff balances. When reading the examples, remember that the payoff balance is always the net of the short-term receivable and the long-term receivable balances.

Booking The Principal Amount Using An Adjustment

When a loan service agreement is activated (i.e., when its status changes from pending start to active), an adjustment is created to book the principal amount. If the customer takes out a loan of 10,000, the adjustment's financial transaction looks as follows:

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan service agreement is activated (and an adjustment is created to book the principal)	Long Term Loan Receivable 10,000 Cash <10,000>	0	10,000	0	10,000

This adjustment is issued if:

- The service agreement's SA type indicates a [special role](#) of Loan.
- The loan service agreement's Total Amount to Bill contains an amount (i.e., the loan amount).
- The loan service agreement was created using a [start option](#) on [Start/Stop Maintenance](#) AND the start option references an [adjustment type](#) and this adjustment type has been set up as follows:
 - The adjustment type's distribution code should reference the GL account to credit (e.g., Cash).
 - The adjustment type's FT algorithm reference $\text{Payoff Amt} = \text{Adj Amt} / \text{Current Amt} = 0$ (booking principal only impacts a customer's payoff balance).

Note that because this financial transaction doesn't have a current amount (the customer doesn't actually owe a current amount yet), there is no need to book anything to the short-term receivables distribution code.

Loan Amortization Schedule Calculation

The amortization schedule is a projection of the amount of principal and interest in each payment over the life of the loan. The amortization schedule may change, for example if the interest rate changes or the customer makes an overpayment (reducing the principal balance).

A loan's amortization schedule is calculated when an operator clicks the **Calculate** button on [Loan - Main](#). Please be aware that when this button is clicked, an algorithm plugged in on the loan [SA type](#) actually calculates the amortization schedule (refer to the algorithm type [LSCH-SI](#) for more information).

Billing For Loans And Interest Calculation

A bill segment is produced for a loan when its service agreement's account is next billed.

Factors on a loan's SA type controls when a bill segment is produced for a loan. Typically SA types for loan service agreements are set up to use anniversary [calendar billing](#). For this configuration:

- You must indicate the type of anniversary billing in the calendar billing option. Currently, we only support the Anniversary Future Billing (meaning that loans are billed in advance just like a classic home loan is). Refer to [Using The Anniversary Method](#) for more information about how this billing method controls the end date of the loan bill segment.
- You must reference an [anniversary billing frequency](#) consistent with the [recurring charge frequency](#) (e.g., monthly, quarterly, etc.).

To set up for a loan that is billed on a monthly basis, you would define the following fields in SA Type - Billing:

- **Use Calendar Billing:** Anniversary Future Billing
- **Anniversary Bill Frequency:** Monthly
- **Total Bill Amount:** Required
- **Recurring Charge:** Required
- **Recurring Charge Frequency:** Monthly
- **Total Amount To Bill Label:** Loan Amount
- **Recurring Chg Amt Label:** Payment Amount

If your type of loan must be billed on an exact date (for example, always on the 15th of the month) or with an exact number of days between each bill (for example, every 14 days), then your loan should be set up to use the calendar billing option of Use Bill Period. In order for your loan bills to be created on specific dates, the system makes the following assumptions:

- Your **bill cycle schedule** for the loan's account is defined with the dates that you want the loan to bill and is defined with the window start date equal to the window end date.
- You define a **bill period** schedule corresponding to your bill cycle schedule. Each bill period schedule's bill date should match your bill cycle window start date and each bill period schedule's bill segment end date should be set to the loan period end date.
- Considerations for the first bill. If the loan SA Type's Initial Start Date Option indicates that the first day of the service agreement should be billed, then the loan SA's start date should match the window start date of the next bill cycle schedule for the account. If the loan SA Type's Initial Start Date Option indicates that first day of the service agreement should not be included, then the loan SA's start date should be one day prior to the window start date of the next bill cycle schedule for the account.
- Define your **recurring charge frequency** to match the frequency of your bill periods.

How the bill segment affects the customer's balance, and how it affects the general ledger are controlled by several algorithms defined on the loan service agreement's **SA type** and **bill segment type**:

- The SA type's loan schedule algorithm controls how the **loan amortization schedule** is calculated.
- The SA type's loan interest charge algorithm controls how interest is calculated.
- The bill segment type's create algorithm controls how the bill lines are constructed.
- The bill segment type's financial algorithm controls how the general ledger is affected by the bill.

The second entry in the following table contains an example of the financial transaction produced by the base package algorithms (note, the first financial transaction in the table was described under [Booking The Principal Amount Using An Adjustment](#)).

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan service agreement is activated	Long Term Loan Receivable 10000 Cash <10000>	0	10,000	0	10,000 long-term: 10,000
First bill segment is produced	Interest: Long Term Loan Receivable 41.66 Interest Revenue <41.66> Transfer Long Term To Short Term: Short Term Loan Receivable 438.71	438.71	41.66	438.71	10,041.66 short-term: 438.71 long-term: 9,602.95

Several important points are illustrated above:

- When a bill segment is produced for a loan service agreement, the following takes place (if you use a bill segment creation algorithm of Create Bill Segment for Loans):
 - The loan SA type's bill segment creation algorithm calls the SA type's loan interest charge algorithm.
 - The base package loan interest charge algorithm calculates simple interest based on: 1) unbilled principal (i.e., the service agreement's payoff balance minus the current balance), 2) the number of billing periods covered by the bill, and 3) the [interest rate](#). Refer to the algorithm type [LINT-SI](#) for more information about the base package interest calculation algorithm.

NOTE:

No interest on interest and no interest on past due amounts. Just like a classic home loan, the base package algorithms do not calculate interest on interest, nor do they calculate interest on past due amounts. If you want to levy a late payment charge, use the SA type's late payment processing. If your organization calculates interest differently, you must develop your own algorithm(s).

- The SA type's bill segment creation algorithm uses the calculated interest to format the bill segment's bill lines. It creates one bill line to show the amount of interest in the payment, and another bill line to show the amount of principal. The principal amount is equal to the service agreement's periodic payment amount minus the amount of calculated interest.
- The financial transaction illustrated above is created if you use a bill segment financial algorithm of $\text{Payoff Amt} = \text{Interest} / \text{Current Amt} = \text{Bill Amount}$ on the loan's bill segment type. The following explains how this algorithm works:
 - The SA's current balance increases by the amount of the loan's periodic payment amount (i.e., its recurring charge amount). In other words, the amount the customer thinks they owe increases by 438.71.
 - The SA's payoff balance increases by the amount of interest. In other words, if the customer wanted to payoff the loan, they'd owe 10,041.66.
 - The *Interest* portion of the GL accounting is straightforward (if you're an accountant). It simply takes the amount of interest and debits it to the SA type's receivable distribution code (long-term receivables) and credits it to the distribution code defined on the interest rate's bill factor (typically interest revenue).
 - The *Transfer* portion of the GL accounting transfers moneys from long-term receivables (i.e., the unbilled principal) to short term receivables (the billed portion of the debt). The amount transferred is equal to the FT's effect on the service agreement's current balance, allowing the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). Remember that the payoff balance is the net of the short-term and long-term receivables.

Paying What Is Owed

When a payment is made for a loan:

- The service agreement's payoff amount is reduced by the payment amount.
- The service agreement's current amount is reduced by the payment amount.

The events that happen when a customer makes a payment against a loan is controlled by the FT algorithm plugged in on the loan service agreement's payment segment type. We'll use an example to help explain how this algorithm works. The 3rd entry in the following table illustrates the financial transaction produced when a payment is made (note, the first two financial transactions were described above).

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan service agreement is activated	Long Term Loan	0	10,000	0	10,000
	Receivable 10000				long-term: 10,000
	Cash <10000>				
First bill segment is produced	Interest:	438.71	41.66	438.71	10,041.66
	Long Term Loan				short-term: 438.71
	Receivable 41.66				long-term: 9,602.95
	Interest Revenue <41.66>				
	Transfer Long Term To Short Term:				
	Short Term Loan Receivable 438.71				
	Long Term Loan Receivable <438.71>				
Payment is made	Affect Cash	-438.71	-438.71	0	9,602.95
	Cash 438.71				short-term: 0
	Long Term Loan Receivable <438.71>				long-term: 9,602.95
	Transfer Long Term To Short Term:				
	Long Term Loan Receivable 438.71				
	Short Term Loan Receivable <438.71>				

The financial transaction illustrated above is created if you use a payment segment FT creation algorithm of $\text{Payoff Amt} = \text{Current Amt} = \text{Pay Amt}$ on the loan's payment segment type. The following explains how this algorithm works:

- The SA's current balance decreases by the amount of the payment amount. In other words, the customer thinks they owe 0 after the payment. Note refer to [Overpayments](#) for information about how an overpayment affects the SA's balances and the general ledger.
- The SA's payoff balance decreases by the amount of the payment. In other words, if the customer wanted to payoff their loan, they'd owe 9,602.95.
- The *Cash* portion of the GL accounting is straightforward (if you're an accountant). It simply takes the amount of the payment and debits it to the payment segment type's distribution code (typically cash) and credits it to the SA type's receivable distribution code (long-term receivable).
- The *Transfer* portion of the GL accounting effectively reduces short-term receivables by the FT's effect on the customer's current balance. This reduction is handled by an offsetting increase to long-term receivables (to make up for reduction made when the cash was applied). Again, this differentiation between short-term and long-term receivables allows the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term).

Overpayments On Loans

You can determine whether you want to accept loan overpayments. Overpayments reduce the principal amount (the amount owed on the loan), which follows the philosophy adopted by a typical home loan.

When the payment is made, any overpayments are distributed according to the overpayment distribution algorithm defined for the customer class. Any customer classes for which you want to allow loan overpayments should use an overpayment distribution algorithm that keeps the overpayment on the loan SA.

When the payment transaction is frozen, the system checks to see if there is a credit amount on the loan SA's current balance. If a credit exists, the customer has made an overpayment and an adjustment is created to zero out the current balance and transfer the amount of the credit from the SA's short-term receivable to long-term receivable. The adjustment may appear on the customer's next bill to show the additional amount paid against the principal.

The algorithm that controls this adjustment to remove the credit on current balance is plugged in on the loan service agreement's SA type and is applied on the SA Type - Payment Freeze system event.

The third and fourth entries in the following table illustrate an overpayment (note, the first two financial transactions were described above).

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan service agreement is activated	Long Term Loan Receivable 10000	0	10,000	0	10,000
	Cash <10000>				long-term: 10,000
First bill segment is produced	Interest:	438.71	41.66	438.71	10,041.66
	Long Term Loan Receivable 41.66				short-term: 438.71
	Interest Revenue <41.66>				long-term: 9,602.95
	Transfer Long Term To Short Term:				
	Short Term Loan Receivable 438.71				
	Long Term Loan Receivable <438.71>				
Payment is made (with an overpayment of 200.00)	Affect Cash	-638.71	-638.71	-200.00	9,402.95
	Cash 638.71				short-term: <200.00>
	Long Term Loan Receivable <638.71>				long-term: 9,602.95
	Transfer Long Term To Short Term:				
	Long Term Loan Receivable 638.71				
	Short Term Loan Receivable <638.71>				
Create adjustment to remove SA's credit.	Transfer Short Term Credit to Long Term:	200.00	0	0	9,402.95
	Short Term Loan Receivable 200.00				short-term: 0
	Long Term Loan Receivable <200.00>				long-term: 9,402.95

In the third financial transaction illustrated above, the billed amount of the payment works essentially the same as that illustrated under [Paying What Is Owed](#). If the Keep Overpayment on Loan SA algorithm is plugged in on the overpayment distribution event on the customer class, the overpayment amount is applied to the loan SA, creating a credit on the SA's current balance. The following explains how this algorithm works:

- If the account has an loan SA and there is an excess credit, the credit is applied to the loan SA (as long as this does not cause the loan SA to have a credit payoff balance).
- If there is not a loan SA to which the credit can be applied, the algorithm checks to see if there is an open excess credit SA for the account.
 - If so, the excess credit amount is applied to the excess credit SA.
 - If not, the algorithm creates an excess credit SA and applies the amount to this SA.

The fourth financial transaction illustrated above is created if the Create Adjustment to Remove SA's Credit algorithm is plugged in on the loan's SA type's payment freeze event. The following explains how this algorithm works:

- If the SA's current balance is less than zero, the algorithm creates a frozen adjustment that removes the credit by transferring the credit from the short-term receivable to the long-term receivable. This adjustment ID is captured on the pay segment so the adjustment can be canceled if the payment is later canceled. Note that the adjustment cancel reason used by the system in this case is specified on the Financial Transaction Options [Feature Configuration](#) using the Adjustment Cancel Reason For Payment Linked To Adjustment option type.
- The SA's current balance increases by the amount of the credit transfer. In other words, the customer thinks they owe 0 after the transfer.
- The SA's payoff balance doesn't change because the payoff balance is always the net of the short-term and long-term receivables. In other words, if the customer wanted to payoff their loan, they'd still owe 9,402.95.

NOTE:

Overpayments and interest. The base package interest calculation algorithm (plugged in on the loan's SA type) does not take into consideration the exact date that the overpayment is made when calculating the interest for the period. It only takes into consideration the outstanding principal amount (payoff balance - current balance) at the time of the interest calculation.

Adjusting Loan Amounts

You would issue ad hoc adjustments if you need to change a loan's payoff and/or current balance outside of the normal billing / payment functions.

An adjustment can:

- Reduce a loan's payoff balance.
- Reduce a loan's current balance (i.e., how much the customer thinks they currently owe).
- Reduce both the loan's payoff and current balance.

For adjustments that affect payoff amount only:

- These adjustments are used to change the principal owed, e.g., if an additional amount is loaned.
- The adjustment's adjustment type should reference the Payoff Amt = Adj / Current Amt = 0 FT algorithm.
- GL lines will be generated to reflect the change to principal.

For adjustments that reduce current amount only:

- These adjustments can be used to change the amount that the customer must pay as part of the next bill.

- The adjustment's adjustment type should reference the Payoff Amt = 0 / Current Amt = Adj Amount (no GL) FT algorithm.
- Typically, GL lines are not generated for FTs that only affect the customer's current balance. However, moneys must be transferred from long to short in the amount of the adjustment (as described above under [Payoff Balance and Current Balance for Loans](#)).

For adjustments that reduce both payoff and current amount:

- These adjustment types can be used to levy additional charges, such as late fees, or to correct interest calculations.
- The adjustment's adjustment type should reference the Payoff Amt = Adj / Current Amt = Adj FT algorithm.
- GL lines are generated to reflect the change to principal. In addition, GL lines must be generated to transfer money from long to short in the amount of the adjustment (as described above under [Payoff Balance and Current Balance for Loans](#)).

NOTE:

Adjustments that affect the principal balance (payoff balance - current balance) affect the term of the loan because interest is based on the principal balance.

Adjustments can cause credit balances to exist. If you credit the loan SA, it is possible for the current balance to become negative. You may need to create additional adjustments that affect the current amount, depending on whether the customer needs to pay this amount as part of the next payment.

Writing Off Loans

Loans are written off using the standard write-off processing.

FASTPATH:

Refer to [The Big Picture Of Write Off Processing](#) for background information.

We illustrate the classic financial transactions that transpire to financially write-off a loan to help illustrate important points (these are the fourth and fifth entries in the following table):

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan service agreement is activated	Long Term Loan	0	10,000	0	10,000
	Receivable 1000				long-term: 10,000
	Cash <1000>				
First bill segment is produced	Interest:	438.71	41.66	438.71	10,041.66
	Long Term Loan				short-term: 438.71
	Receivable 41.66				long-term: 9,602.95
	Interest Revenue <41.66>				
	Transfer Long Term To Short Term:				
Payment is made (with an	Short Term Loan				
	Receivable 438.71				
	Long Term Loan Receivable <438.71>				
	Affect Cash	-638.71	-638.71	-200.00	9,402.95

overpayment of 200.00)	Cash 638.71				short-term: <200.00>
	Long Term Loan Receivable <638.71>				long-term: 9,602.95
	Transfer Long Term To Short Term:				
	Long Term Loan Receivable 638.71				
	Short Term Loan Receivable <638.71>				
Create adjustment to remove SA's credit.	Transfer Short Term	200.00	0	0	9,402.95
	Credit to Long Term:				short-term: 0
	Short Term Loan Receivable 200.00				long-term: 9,402.95
	Long Term Loan Receivable <200.00>				
Sync up current with payoff balance	Transfer Long Term To Short Term:	9,402.95	0	9,402.95	9,402.95
	Short Term Loan Receivable 9,402.95				short-term: 9,402.95
	Long Term Loan Receivable <9,402.95>				long-term: 0
Transfer balance to a write-off SA	Transfer From Loan SA:	-9,402.95	-9,402.95	0	0
	Write Off Xfer Clearing 9,402.95				
	Long Term Loan Receivable <9,402.95>				
	Long Term Loan Receivable 9,402.95				
	Short Term Loan Receivable <9,402.95>				
	Transfer To Write Off SA:				
	Bad Loans Expense 9,402.95				
	Write Off Xfer Clearing <9,402.95>				
	Note, these will cause a balance of 9,402.55 to exist on the write-off service agreement.				

The only unusual portion of the last two financial transactions is the impact on short and long term receivables. Please see the examples above under [Billing For Loans And Interest Calculation](#) and [Paying What Is Owed](#) to understand how any impact to a loan's current balance causes this type of financial transfer to occur.

Distribution Codes for Loans

As explained above under [Payoff Balance and Current Balance for Loans](#), loans have two accounts receivable distribution codes: long term and short term. These two codes allow the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). Both receivables distribution codes are defined on the loan SA type.

In addition, loans have a distribution code used to book interest revenue. The interest revenue distribution code is defined on the loan's bill factor value for a revenue class (defined on the loan's SA type). For example, on the bill factor you can use one distribution code to book interest revenue from the residential revenue class and another distribution code to book interest revenue from the commercial revenue class. In this example, you create two loan SA types, one for residential revenue and the other for commercial revenue.

Loans may also have a bad loan debt (expense) distribution code that is used when writing off a loan. The bad loan debt distribution code is defined on the loan's write-off service agreement type. Refer to [Defining Credit & Collections Options](#) for more information.

Setting Up The System To Enable Loans

The above topics provided background information about how loans are supported in the system. The following discussion summarizes the various setup tasks alluded to above.

Distribution Code

You must set up the following [distribution codes](#):

- Long term receivables
- Short term receivables
- Interest revenue
- Bad loan debt

Adjustment Types

The following adjustment types are needed:

- Activate a loan. This should reference a distribution code associated with cash or the cash equivalent and an FT algorithm of $\text{Payoff Amt} = \text{Adj} / \text{Current Amt} = 0$.

NOTE:

Creating Checks for Loan Amounts. If you want the system to initiate a check to the customer for the loan amount, the loan activation adjustment type should indicate an **A/P Request Type Code**. Refer to [Controls The Interface To A/P & 1099 Reporting](#) for more information.

- Remove Credit (Overpayment). This adjustment should reference an FT algorithm of $\text{Payoff Amt} = 0 / \text{Current Amt} = \text{Adj Amount}$ (no GL). Even though this algorithm indicates that there is no effect on the GL, there is a special exception built in for loan SAs. The special exception creates GL details to transfer the current balance (short-term receivable) to the long-term receivable. Refer to [Overpayments](#) for more information.

NOTE:

Printing Overpayments on Bills. If you want the overpayment adjustment to appear on customers' bills, turn on **Print by Default** and enter a **Description on Bill** (e.g. "Additional Principal"). For more information, refer to [Controls Information Printed On The Bill](#).

- Adjustment types to perform any of the adjustments described under [Adjusting Loan Amounts](#).

Adjustment Type Profile

Create an adjustment type profile that references the adjustment types used on a loan. Besides the above adjustment types, it should also reference adjustment types to levy late payment charges (if applicable), levy non-sufficient funds charges (if applicable), refund overpayments (if applicable), sync current balance with payoff balance at write-off time, transfer balances to a write-off service agreement, and write-down small balances.

Algorithms

Add the following [algorithms](#):

- Overpayment Distribution. This algorithm is later plugged in on the customer class for the Overpayment Distribution system event. Refer to the algorithm type [OVPI-LO-CSA](#) for more information about the base package algorithm.
- Bill Segment Creation for Loans. This algorithm is later plugged in on the loan's bill segment type. Refer to the algorithm type [BSBS-LO](#) for more information about the base package algorithm.
- Bill Segment Financial Transaction Creation for Loans. This algorithm is later plugged in on the loan's bill segment type. Refer to the algorithm type [BSBF-LO](#) for more information about the base package algorithm.
- Amortization Schedule. This algorithm is later plugged in on the SA type for the Loan Schedule system event. Refer to the algorithm type [LSCH-SI](#) for more information about the base package algorithm.
- Interest Calculation. This algorithm is later plugged in on the SA type for the Loan Interest Charge system event. Refer to the [LINT-SI](#) algorithm type for more information about the base package algorithm.
- Payment Periods/Payment Amount Calculation. This algorithm is later plugged in on the SA type for the Loan Periods and Amount system event. Refer to the algorithm type [LPDA-SI](#) for more information about the base package algorithm.
- Loan SA Payment Freeze. This algorithm has a parameter that must reference the Remove Credit adjustment type defined above. This algorithm is later plugged in on the SA type for the Payment Freeze system event. Refer to the algorithm type [STPZ-RMVCR](#) for more information about the base package algorithm.

Bill Factor

You must set up a [bill factor](#) that defines the interest rate. If you have different interest rates for different types of loans, you can create a separate bill factor for each or you can use start options to override the interest rate.

On the bill factor's [bill factor value](#), make sure to reference the GL distribution code used to book interest revenue for the revenue class specified on the loan's SA type.

Customer Class

Any [customer class](#) on which you want to allow overpayments for a loan must use the overpayment distribution algorithm defined above. The overpayment distribution algorithm keeps the overpayment on the loan SA rather than transferring it to an excess credit SA, allowing a subsequent adjustment to apply the overpayment to the principal balance.

Bill Segment Type

Create a bill segment type that references the bill segment creation algorithm defined above and the FT creation algorithm defined above.

Frequency

Create [frequency](#) codes to correspond to the frequency of your loans. When setting up your SA types, you must indicate a recurring charge frequency and, if you use the Anniversary Billing Option, you must indicate an anniversary frequency.

Bill Period

If you use the Use Bill Period option, set up a bill period with an appropriate schedule of dates for billing your loan.

Bill Cycle Schedule

If you use the Use Bill Period option, the bill cycle schedule for these types of loans should be defined with an appropriate schedule of dates for billing your loan.

Collection, Severance and Write Off Processes

You should set up the appropriate credit and collections information. Refer to [Defining Credit & Collections Options](#) for more information.

SA Type

You must set up a [SA type](#) for your loan service agreements (you may need multiple SA types if you have different business rules for different types of loans). The following points describe the minimal requirements for a loan SA type.

SA Type - Main

Define the following options:

- **Distribution Code** should be a long-term receivable code.

- **Service Type** should reference something like "Miscellaneous Service".
- Specify the **Revenue Class** that, together with the interest bill factor, determines the distribution code used to book the loan interest revenue.
- **Start Option** should be turned on.
- The Payment Segment Type should reference the Normal Payment. The base package payment segment financial algorithm (Payoff Amt = Current Amt = Pay Amt) used for Normal Payment pay segment types creates the additional GL details to transfer the credit from long-term receivables to short-term receivables if the SA's special role is Loan.
- Turn on **Late Payment Charge** if applicable.
- Define an appropriate **Adjustment Type (Synch Current)** that will cause current balance to be synchronized with payoff balance (if the loan is [written off](#)).

SA Type - Detail

Define the following options:

- **Special Role** is Loan.
- Specify the **Interest Bill Factor** set up above. You can use start options to override.
- Use **Override Interest Flag** to indicate whether the interest rate defined on the interest bill factor may be overridden at the SA level. If you select Allowed, the interest rate may be overridden by a contract value on a start option or the SA.
- Use the short-term receivable account defined above as the **Loan A/R Distribution Code**. If you do not want the system to differentiate between short-term receivables and long-term receivables, make the loan A/R distribution code the same as the distribution code (above).

SA Type - Billing

FASTPATH:

For an overview of some of these options, be sure to refer to [Billing For Loans And Interest Calculation](#) for more information.

Define the following options:

- The [Bill Segment Type](#) should reference the value created above.
- **Characteristic Premise Required** should not be checked. (A loan SA is not a premise-based service. SA types that have this box checked are filtered out of the SA type search when the start method is Start a SA, so users will never be able to start a loan SA that requires a characteristic premise.)
- **Use Calendar Billing** must equal Anniversary Future Billing or Use Bill Period.
- **Bill Period** is required for the Use Bill Period option.
- **Anniversary Billing Frequency** is required for the Anniversary Future Billing option and must equal the periodicity of the bills (monthly, weekly, etc.).
- **Total Bill Amount** is required (it holds the principal).
- **Total Amount To Bill Label** should reference something like "Loan Amount".
- **Recurring Charge** is required (it holds the periodic payment amount).
- **Recurring Chg Amt Label** should reference something like "Payment Amount".

- **Recurring Charge Frequency** is required (it defines the periodicity associated with the recurring charge amount) and must be the same as the bill period frequency or the anniversary billing frequency (based on your **Use Calendar Billing** option).

SA Type - Rate

Turn off the **Rate Required** switch as loans do not use rates.

SA Type - SP Type

Turn off the **Service Points Required** switch as loans do not have service points.

SA Type - Adjustment Profile

Adjustment Type Profile should reference the profile set up above.

SA Type - Credit and Collections

The credit and collections information should reference a **Severance Process Template** that simply expires the loan. The **Debt Class** and **Write Off Debt Class** should reference an appropriate value consistent with your credit and collections rules. Refer to [Defining Credit & Collections Options](#) for more information.

SA Type - Algorithms

The Loan Schedule, Loan Interest Charge, Loan Periods, and Amounts and Payment Freeze [algorithms](#) defined above must be set up.

Start Options

Loans require a start option to define the adjustment type that is used to create the adjustment to book the initial principal amount when the loan is activated. Loan start options can also specify default values for loan amount, payment amount, and number of periods.

Create at least one start option for each loan SA type. The following information should be defined:

- **Adjustment Type** should reference the adjustment type defined above to book principal.
- **Recurring Charge Amount** (Payment Amount) should only be defined if you have standard loan payments.
- **Total Amount To Bill** (Loan Amount) should only be defined if you have standard loan amounts.
- **Number of Payment Periods** should be the number of payment periods in the loan (if you have a standard loan period).
- If you want to define a set of standard APRs for a loan SA Type, set up a **Contract Value** on the Rate Info page. The **Bill Factor** should match the interest bill factor specified on the [SA Type](#).

Defining Non-billed Budget Options

A non-billed budget (NBB) is a payment plan that allows your customers to pay set amounts at specified intervals (e.g. every two weeks). Non-billed budgets are typically used when your company bills on an infrequent basis and you want to provide your customers with a mechanism to make smaller payments more frequently. As the name suggests, bills are not created for the non-billed budget's scheduled payments; customers must remember to make their payments at the scheduled intervals. The topics in this section describe how to design and set up non-billed budgets.

NOTE:

Non-billed budgets are optional. The system configuration requirements described in this section are only relevant if your organization offers non-billed budgets.

What Is A Non-billed Budget?

A non-billed budget is a special type of budget or payment plan that encompasses three major elements:

- A set of scheduled payments
- The business rules used to recommend and potentially renew the payment schedule
- The business rules that govern the financial impact on the current and payoff balances of the SAs covered by the payment schedule

Non-billed budgets are managed via a service agreement whose SA type has a special role of Non-billed Budget. If an SA type has a special role of non-billed budget it must have one or more recommendation rules, and SAs of that type are allowed to have payment schedules.

The Financial Impact Of Non-billed Budgets

When you set up the non-billed budget SA type, you can indicate whether the non-billed budget is monitored by the [account debt monitor](#) process. The following sections contain examples of financial transactions for non-billed budgets that are monitored by the account debt monitor.

NOTE:

Unmonitored non-billed budgets. SAs that are covered by unmonitored non-billed budgets are subject to different financial treatment than those that are monitored. The [financial transactions relevant for unmonitored non-billed budgets](#) are discussed later.

Current Amount For SAs Covered By A Non-billed Budget

As described under [Current Amount versus Payoff Amount](#), the values of the current balance and payoff balance are the same for most financial transactions. One exception is for SAs that are covered by a non-billed budget in which case the current balance is always zero and the payoff balance is always the amount the customer really owes.

When a non-billed budget is activated or an SA is added to a non-billed budget, the system creates adjustments for all affected SAs to set the current amount equal to zero. The adjustment type that references the Payoff Amt = 0 / Current Amt = Adj Amount (no GL) algorithm is taken from the SA's SA type.

NOTE:

Non-billed Budgets and Open Item Accounts. If the non-billed budget is for an open item account, no match event is created for the financial transaction that reduces the current amount to zero. This FT must be manually matched if required.

The following example shows the effect of activating a non-billed budget that covers an electric SA with a current balance of 25. The transactions for the electric SA are illustrated on the right and the transactions for the non-billed budget SA are illustrated on the left.

Event	Effect On	Effect On	Current	Payoff	Effect On	Effect On	Current	Payoff
	Current	Payoff	Balance	Balance	Current	Payoff	Balance	Balance
	Balance	Balance			Balance	Balance		
	Electric SA				Non-billed Budget SA			
Starting balance	0	0	25	25				
Non-billed budget is activated	-25	0	0	25				

By setting the current balance for the covered SAs to zero, the SAs are insulated from the regular debt monitoring process. Depending on the non-billed budget [recommendation algorithm](#), the payoff balance can be ignored, divided evenly between the scheduled payments or added to the first scheduled payment.

Activating a non-billed budget has no affect on its own current or payoff balances.

Scheduled And Actual Payments On The Non-billed Budget

When a scheduled payment is due, an adjustment is created to increase the non-billed budget's current balance by the expected amount. The current balance on the SA can be monitored to ensure that payments are being made on time.

When the payment is made, the non-billed budget's current balance is reduced to zero and the non-billed budget's payoff balance reflects the accumulated credit from the payment. This accumulated credit is transferred to the covered SAs when the next bill for the account is completed.

The following example illustrates scheduled and actual payments for the non-billed budget in the previous example. The amount of the scheduled non-billed budget payments is 10. Note that the first two transactions were described above.

Event	Effect On	Effect On	Current	Payoff	Effect On	Effect On	Current	Payoff
	Current	Payoff	Balance	Balance	Current	Payoff	Balance	Balance
	Balance	Balance			Balance	Balance		
	Electric SA				Non-billed Budget SA			
Starting balance	0	0	25	25				
Non-billed budget is activated	-25	0	0	25				
First scheduled payment is due					10	0	10	0

Payment	-10	-10	0	-10
---------	-----	-----	---	-----

An algorithm (Process NBB Scheduled Payment) plugged in on the non-billed budget SA type creates the appropriate financial transactions when the scheduled payment is due. The algorithm is called by the Non-billed Budget Scheduled Payment Processing ([NBBPS](#)) background process.

The normal payment processing handles the adjustments created when the payment is made.

NOTE:

Non-billed Budget Payment Cancellation. If a payment is canceled, the financial transaction is reversed.

Automatic Payments And Non-billed Budgets. Users may set up a non-billed budget to use [automatic payments](#) by setting up the account's auto pay options. Users may also exclude the non-billed budget from automatic payment if the account is set up for automatic payment. The [NBB Scheduled Payment Automatic Payment Create](#) background process calls the [Auto Pay Creation](#) algorithm to create the auto payment for a non-billed budget.

Overpayments for Non-billed Budgets

Typically, payments in excess of the non-billed budget's current balance are credited to an overpayment (excess credit) SA. When the next adjustment is created for a scheduled payment, the credit on the overpayment SA is used to relieve the non-billed budget current balance.

NOTE:

Overpayment Distribution Algorithm. The overpayment distribution is a function of the overpayment distribution algorithm plugged in on the account's customer class. We strongly recommend that the non-billed budget SA type be set up so that overpayment is not allowed. Any excess payments should go to an overpayment SA. For more information about overpayment distribution, refer to [Overpayment Segmentation](#).

In the example below, the customer pays 20 for a scheduled payment instead of 10. The non-billed budget SA is illustrated on the right and the overpayment SA is illustrated on the left. The electric SA is not illustrated in the example below because scheduled payments, payments and overpayments have no effect on covered SAs. The first four transactions were illustrated above.

Event	Effect On		Current Balance	Payoff Balance	Effect On		Current Balance	Payoff Balance
	Current Balance	Payoff Balance			Current Balance	Payoff Balance		
	Non-billed Budget SA				Overpayment SA			
Starting balance								
Non-billed budget is activated								
First scheduled payment is due	10	0	10	0				
Payment	-10	-10	0	-10				
Second scheduled	10	0	10	-10				

payment is due								
Over-payment	-10	-10	0	-20	-10	-10	-10	-10
Third scheduled payment is due	10	0	10	-20				
Transfer Adjustment	-10	-10	0	-30	10	10	0	0

After the Process NBB Scheduled Payment algorithm creates the next scheduled payment, it looks for a credit amount on the overpayment SA and creates an adjustment to transfer the credit balance (or the amount of the payment if the credit is more than the scheduled payment amount) from the overpayment SA to the non-billed budget SA. The overpayment transfer adjustment type and the overpayment SA type are specified as parameters to the algorithm.

Underpayments For Non-billed Budgets

An insufficient payment or a canceled payment leaves a current balance on the non-billed budget SA.

In the example below, the customer makes a payment of 7 for the fourth scheduled payment. The example below does not show the overpayment SA (illustrated above). The first eight transactions are discussed above.

Event	Effect On	Effect On	Current Balance	Payoff Balance	Effect On	Effect On	Current Balance	Payoff Balance
	Current Balance	Payoff Balance			Current Balance	Payoff Balance		
	Electric SA				Non-billed Budget SA			
Starting balance	0	0	25	25				
Non-billed budget is activated	-25	0	0	25				
First scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10
Second scheduled payment is due					10	0	10	-10
Over-payment					-10	-10	0	-20
Third scheduled payment is due					10	0	10	-20

Transfer Adjustment	-10	-10	0	-30
Fourth scheduled payment	10	0	10	-30
Underpayment	-7	-7	3	-37

The Process NBB Scheduled Payment algorithm creates a trigger to ensure that the current balance is monitored by the account debt monitor. Refer to [Credit And Collections And Non-billed Budgets](#) for more information.

Billing For SAs Covered By The Non-billed Budget

When the next bill for the account is completed, the credit on the non-billed budget is transferred to the covered SAs. The credit is prorated over the covered SAs according to the relative payoff balances on each SA.

In the example below, the electric SA's bill is 33. The first ten transactions are discussed above.

Event	Effect On	Effect On	Current Balance	Payoff Balance	Effect On	Effect On	Current Balance	Payoff Balance
	Current Balance	Payoff Balance			Current Balance	Payoff Balance		
	Electric SA				Non-billed Budget SA			
Starting balance	0	0	25	25				
Non-billed budget is activated	-25	0	0	25				
First scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10
Second scheduled payment is due					10	0	10	-10
Over-payment					-10	-10	0	-20
Third scheduled payment is due					10	0	10	-20
Transfer Adjustment					-10	-10	0	-30
Fourth scheduled payment					10	0	10	-30
Underpayment					-7	-7	3	-37

Bill	0	33	0	58				
Bill completion (transfer adjustment)	0	-37	0	21	0	37	3	0

When a bill segment financial transaction is created, the current amount is set to zero for SAs that are covered by the non-billed budget (if you use a bill segment FT algorithm of $\text{Payoff Amt} = \text{Bill Amt} / \text{Current Amt} = \text{Amt Due}$). Though not evident by the name, the $\text{Payoff Amt} = \text{Bill Amt} / \text{Current Amt} = \text{Amt Due}$ algorithm does set the current amount to zero for monitored non-billed budgets. (For SAs with other roles, the current amount is equal to the amount due or the recurring charge.)

A bill completion algorithm transfers money from the non-billed budget to the covered SAs (if you plug in the NBB Credit Transfer bill completion algorithm on the non-billed budget SA type). The algorithm type supplied with the base package distributes the credit using the method described in [Distributing Non-billed Budget Credit](#).

NOTE:

Canceled Bill Segments. No new processing occurs when a bill segment is canceled; any credit balance remains on the covered SA.

Distributing Non-billed Budget Credit

Both the NBB Credit Transfer bill completion algorithm type and the non-billed budget SA stop algorithm type supplied with the base package use the same method of distributing a credit from a non-billed budget SA to the covered SAs. The following points describe how the credit is distributed:

- Covered SAs that are already in credit (due to some other circumstance, such as a cancellation and rebill) are excluded from the distribution.
- The distribution to each covered SA will not exceed its total payoff to ensure that none of the covered SAs have a credit balance.
- The credit is prorated over the covered SAs according to the relative payoff balances on each SA.
- The calculation of the payoff balance is adjusted to exclude the current balance to ensure that the credit is prorated over the debt covered by the budget, not any ad-hoc debt for the SA.

NOTE:

The current balance on covered SAs should always be zero. The only exception occurs if an adjustment has been added that directly affected the SA balance. In this case, the distribution assumes that the balance is outside the non-billed budget and needs to be paid separately.

- Any excess credit remains on the non-billed budget SA until the next distribution takes place or until the [non-billed budget SA is stopped](#).
- The type of adjustments created is determined by the **Adjustment Type (Xfer)** specified on the non-billed budget SA type.

The examples in the table below illustrate the points above.

NBB SA Credit	SA 1 Payoff	SA 1 Current	SA 2 Payoff	SA 2 Current	SA 1 Credit Xfer	SA 2 Credit Xfer
-100.00	-100.00	0.00	-200.00	0.00	0.00	0.00
-100.00	150.00	0.00	-50.00	0.00	-100.00	0.00

-300.00	150.00	0.00	50.00	0.00	-150.00	-50.00
-100.00	150.00	0.00	250.00	0.00	-37.50	-62.50
-100.00	150.00	0.00	250.00	50.00	-42.86	-57.14

The first example above shows that covered SAs with a credit balance are excluded from the distribution. Any excess credit remains on the non-billed budget.

The second example shows that one covered SA has a credit balance, so the entire credit is distributed to the remaining SA.

The third example shows the amount distributed to a covered SA does not exceed its payoff balance. Again, any excess credit remains on the non-billed budget SA.

The fourth example illustrates how the credit is prorated based on the payoff balance. The fifth example illustrates the same prorating but with a current balance on one of the SAs (SA 2). (Remember that the prorating excludes any current balance.)

The prorated amount is calculated by subtracting the current balance from the payoff balance then multiplying the result by the distribution amount and dividing by the total payoff owing of all covered SAs.

Stopping an SA Covered By a Non-billed Budget

If a service agreement covered by a non-billed budget is stopped, the system must bring the current balance and payoff balance of the covered SA back in synch. The system creates an adjustment using the Synch Current adjustment type from the SA's SA type.

In the example below, the electric SA's payoff balance is 21. The first twelve transactions are discussed above.

Event	Effect On	Effect On	Current	Payoff	Effect On	Effect On	Current	Payoff
	Current	Payoff			Current	Payoff		
	Balance	Balance			Balance	Balance		
	Electric SA			Non-billed Budget SA				
Starting balance	0	0	25	25				
Non-billed budget is activated	-25	0	0	25				
First scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10
Second scheduled payment is due					10	0	10	-10
Over-payment					-10	-10	0	-20
Third scheduled payment is due					10	0	10	-20

Transfer Adjustment					-10	-10	0	-30
Fourth scheduled payment					10	0	10	-30
Underpayment					-7	-7	3	-37
Bill	0	33	0	58				
Bill completion (transfer adjustment)	0	-37	0	21	0	37	3	0
SA is stopped	21	0	21	21				

NOTE:

In addition to synching the current and payoff balance, the SA being stopped is removed from the covered SAs collection. When the last covered SA is removed from the collection, the non-billed budget is also [stopped](#).

Financial Transactions For Unmonitored Non-billed Budgets

Some companies have the concept of non-billed budgets where the payments made by the customer are optional. This functionality is implemented as an unmonitored non-billed budget. Unmonitored non-billed budgets allow a customer to make optional prepayments towards a bill.

As explained previously, unmonitored non-billed budgets receive different financial treatment than monitored non-billed budgets. The financial transaction algorithms use the **Non-billed Budget Monitor** flag on the non-billed budget's SA type to create the appropriate financial transactions. The following table describes the differences in the financial treatment of monitored and unmonitored non-billed budgets.

System Event	Monitored Non-billed Budgets	Unmonitored Non-billed Budgets
When a non-billed budget is activated or an SA is added to a non-billed budget...	The system creates adjustments for all affected SAs to set the current balance equal to zero.	The current balances for covered SAs are not zeroed. The activate SA and non-billed budget maintenance processing do not create any adjustments if the non-billed budget is unmonitored.
When a scheduled payment is due...	An adjustment is created to increase the non-billed budget's current balance by the expected amount.	There is no change to the non-billed budget's current balance; it is always zero. The scheduled payment processing background process does not execute the NBB process scheduled payment algorithm .
When the payment is made...	The non-billed budget's current balance is reduced to zero and the non-billed budget's payoff balance reflects the accumulated credit from the payment.	Typically the payments are distributed to an excess credit (overpayment) SA. Refer to Distributing Payments for Unmonitored Non-billed Budgets for more information.
When a bill segment financial transaction is created...	The current amount is set to zero for SAs that are covered by the non-billed budget.	The covered SAs' current amounts are equal to their payoff amounts.

At bill completion...	Money from the non-billed budget is transferred to the covered SAs. Customers may receive a bill for information purposes, but they are not required to pay it.	Money from the overpayment SA is transferred to the account's SAs. The customer is still liable to pay the outstanding balance.
When the non-billed budget is stopped or an SA is removed from a non-billed budget...	The system creates adjustments for all affected SAs to synchronize their current balances with their payoff balances, thus removing the zero current balance and replacing it with the actual current balance.	There is no change to the current balances for covered SAs as the balances already reflect the actual current balance.

Unmonitored non-billed budgets also receive different credit and collections treatment. Refer to [Credit and Collections and Unmonitored Non-billed Budgets](#) for more information.

Distributing Payments for Unmonitored Non-billed Budgets

For non-billed budgets, payments are distributed according to the payment and overpayment algorithms on the customer class. The base package payment distribution algorithm applies a payment first towards any SAs that have overdue or current balances (refer to [Distributing A Payment](#) for more information). Since the unmonitored non-billed budget SA doesn't have a current balance, it is not considered by the payment distribution algorithm. If there aren't any SAs with a current balance, the overpayment distribution algorithm handles the remaining credit. You can elect to:

- Apply the overpayment to an excess credit SA. This is the method that we strongly recommend because all financial transactions are then a function of the normal payment, overpayment and billing processes.
- Apply the overpayment to the highest priority SA that is eligible for overpayment (as specified on the SA type). You can use this method to apply the overpayment to the unmonitored non-billed budget SA. If you use this method, you must also set up the system to [transfer the credit from the unmonitored non-billed budget](#).

NOTE:

Use An Excess Credit SA. We strongly recommend that payments for unmonitored non-billed budgets are distributed to an excess credit SA. In this case, the non-billed budget is just a shell to hold the covered SAs and recommend a payment schedule; all financial transactions are a function of the normal payment, overpayment and billing processes.

FASTPATH:

Refer to [Overpayment Segmentation](#) for a detailed discussion of overpayment distribution options.

Transferring Credit from Unmonitored Non-billed Budgets

If you distribute an overpayment to an unmonitored non-billed budget SA (i.e. the unmonitored non-billed budget maintains a credit balance instead of an overpayment SA), you must plug-in a bill completion algorithm on the SA type to transfer the credit balance to the covered SAs at bill completion time. The bill completion algorithm type ([BCMP-NB](#)) supplied with the base package transfers the credit balance to the covered service agreements when the bill is completed. Additionally, the **Adjustment Type (Xfer)** on unmonitored non-billed budget SA types should reference a FT algorithm of $\text{Payoff Amt} = \text{Adj} / \text{Current Amt} = \text{Adj}$ to ensure that the credit is removed from both the current and payoff balances.

WARNING:

You must create your own SA stop algorithm type for correctly stopping an unmonitored non-billed budget that maintains a credit balance. The SA stop algorithm that is supplied with the base package does NOT transfer remaining credit from the unmonitored non-billed budget SA. (The base package SA stop algorithm transfers the remaining credit

using the overpayment distribution algorithm on the customer type, which you have set up to transfer to back to the unmonitored non-billed budget.)

Designing Non-billed Budgets

The topics in this section describe functionality that you must consider when designing non-billed budgets.

Making SAs Eligible For Non-billed Budgets

A billable SA may be covered by a non-billed budget when its SA type is flagged as Eligible for Non-billed Budget.

All SAs that are eligible for non-billed budgets should reference a bill segment type that uses the $\text{Payoff Amt} = \text{Bill Amt} / \text{Current Amt} = \text{Amt Due}$ bill segment FT algorithm. This algorithm sets the SA's current amount to zero if it is covered by a monitored non-billed budget.

A list of the SAs covered by a non-billed budget is maintained with the non-billed budget SA. This list is used at bill completion to determine the financial transactions that should occur.

Designing Recommendation Rules

Users (and the renewal process) ask the system to recommend the scheduled payments for a non-billed budget. In general, this recommendation process must establish:

- The amount to be paid
- The dates on which the payments are due

We envision many different types of recommendation rules. For example:

- Recommend 26 scheduled payments to be made on a fortnightly basis that are due on Tuesdays.
- Recommend monthly payments that are due on the nearest workday after the 10th of the month.
- Recommend scheduled payments where the customer pays their annual charges in 10 out of 12 months where the payments are not due in November and December.
- Recommend bi-monthly payments where the payments are due on the first workday following the 5th and 20th of the month.

Additionally, the recommendation rules must determine how to handle any outstanding payoff balances for covered SAs. The true-up rule provided with the base package [payment schedule algorithm type](#) can ignore the payoff balance, divide the payoff balance evenly between the scheduled payments, or add the payoff balance to the first scheduled payment.

A recommendation rule comprises three elements:

- An algorithm to calculate an average daily amount (the [NBDA-DA](#) algorithm type provided with the base package uses premise billing history)

NOTE:

Calculating an Amount for Non-utility SAs. The algorithm type supplied with the base package only handles service-point oriented SAs. For example, the average daily amount algorithm calculates an average daily amount. For non-utility SAs, you must develop the appropriate algorithm types.

- Two algorithm types are provided with the base package to calculate a schedule of payments. [NBPS-MON](#) calculates a monthly schedule and [NBPS-PS](#) calculates a scheduled based on a specified number of days.
- A collection of default parameter values for the payment schedule algorithm type

A recommendation rule (based on the algorithm types provided with the base package) is illustrated below.

Rule: Weekly		
Avg Daily Amt Alg: Use premise history		
Pay Schedule Alg Type: Number of days		
Parameter	Value	Override
Number of days in period	7	Y
Number of payments	52	N
True-up rule	Spread	N

Weekly Payments Recommendation Rule

NOTE:

Additional Parameters. Not all of the parameters associated with the weekly payment schedule algorithm type are illustrated. Refer to the [NBPS-PS](#) algorithm type for a detail description of the parameters.

The default parameter values for the payment schedule algorithm type may change over time, so the collection contains an effective date. If default values are changed, these changes do not affect non-billed budgets already in effect. Existing non-billed budgets keep the parameter values that were used when the non-billed budget was started.

A user may override the default parameter values for the payment schedule algorithm type to customize the schedule if an override is allowed for a parameter. Additionally, a user may edit the payment schedule details at any time (provided the payment has not yet been processed).

The parameter values used for the recommendation rule are kept with the non-billed budget SA, so that any customized parameter changes can be re-applied to a renewed non-billed budget. For example, the parameter that determines the payment due day may default to the first of the month. To customize the schedule, this value may be changed to the fifth of the month. This amended value is kept with the non-billed budget SA to ensure that the renewed budget follows the same monthly schedule.

NOTE:

Normally parameter values for an algorithm type are kept with the algorithm. Because the parameters may vary for each non-billed budget, the parameter values are kept with the SA in the case of non-billed budgets.

Refer to [Non-billed Budgets Recommendation Rule - Main](#) for information on creating recommendation rules.

Example Recommendation Rules

The following examples may be helpful in designing and implementing your recommendation rules.

NOTE:

Developing Your Own Payment Schedule Algorithms. The base package comes supplied with a [monthly payment schedule algorithm type](#). You can use this algorithm as an example when creating payment schedule algorithm types for your implementation.

Fortnightly Payments Recommendation Rule Example

The following diagram illustrates a recommendation rule where the customers pay every two weeks. The current balance for any covered SAs is added to the first payment.

Rule: Fortnightly		
Avg Daily Amt Alg: Use premise history		
Pay Schedule Alg Type: Number of days		
Parameter	Value	Override
Number of days in period	14	Y
Number of payments	26	N
True-up rule	Spread	N

Fortnightly Payments Recommendation Rule

Monthly Payments Recommendation Rule Example

The following diagram illustrates a recommendation rule where the customers pay twice monthly on the first of the month. The current balance for any covered SAs is spread out over the scheduled payments.

Rule: Monthly		
Avg Daily Amt Alg: Use premise history		
Pay Schedule Alg Type: Monthly		
Parameter	Value	Override
Day of month	1	Y
Number of payments	12	N
True-up rule	Spread	N

Monthly Payments Recommendation Rule

Ten Out of Twelve Months Recommendation Rule Example

The following diagram illustrates a recommendation rule where the customers pay one a month except for months during a holiday season (November and December). The current balance for any covered SAs is added to the first payment.

Rule: 10 of 12		
Avg Daily Amt Alg: Use premise history		
Pay Schedule Alg Type: X out of Y months		
Parameter	Value	Override
Total number of months	12	N
Months with no payment	11,12	Y
True-up rule	1 st pay	N

Ten Out of Twelve Months Recommendation Rule

Activating Non-billed Budgets

You can plug in an algorithm (of type [SACR-AT](#)) on the SA Creation system event on the non-billed budget SA type to automatically activate the non-billed budget (i.e. transition it from pending start to active status). If you don't use a SA Creation algorithm to activate the non-billed budget, it is activated the next time the [SA activation background process](#) runs.

When a non-billed budget is activated, you can perform special processing using an algorithm plugged in on the SA Activation system event on the non-billed budget SA type. The special processing can be developed to do anything that you would like, for example you could:

- Create a customer contact that with an appropriate letter template can generate a letter to inform the customer of their payment amount and payment schedule.
- Initiate the creation of a payment coupon book for a customer.

The system comes supplied with a sample algorithm type (called [SAAT-CC](#)) that simply creates a customer contact to indicate that the non-billed budget is activated.

Renewing Non-billed Budgets

A non-billed budget can be renewed either manually or via a background process. When the non-billed budget SA is created, the expiration date, renewal date and the recommendation rule used to create the initial budget are kept with the SA. A renewal flag on the non-billed budget SA type controls if a renewal is required, optional or not allowed. If renewal is required, a user must specify a renewal date when creating the service agreement. The renewal date is defaulted on to an SA based on the valued of the **Days Before Expiration for Renewal** field on the SA type.

An algorithm on the SA type can customize the processing required to renew an SA.

The [SA renewal background process](#):

- Executes the SA renewal algorithm (specified on the SA type) when the renewal date is reached (i.e., it is on or before the process date). The base package comes with an algorithm type ([SARN-NB](#)) that determines the current recommendation rule for a non-billed budget and executes the associated payment schedule algorithm using the non-billed budget SA-specific parameter values to generate a new schedule. It returns new expiration and renewal dates.
- If the renewal algorithm is successful, the renewal and expiration date fields on the SA are updated with the new values.
- If the renewal process is not successful, a To Do list entry (of type [TD-SARN](#)) is created for the account and SA.

The new payment schedule that is returned from the renewal process for a non-billed budget is appended to the current schedule.

A user can manually launch the renewal process for a non-billed budget SA by clicking the **Renew NBB** button on the [non-billed budget maintenance page](#).

Expiring Non-billed Budgets

Non-billed budget service agreements may specify an expiration date. The [SA expiration background process](#) initiates the stop process for all pending start or active SAs where the expiration date is reached (before or on the process date).

Stopping Non-billed Budgets

When a non-billed budget stop is initiated, either on request or because it has expired and is not being renewed, the non-billed budget is transitioned to pending stop status. You can plug in an algorithm (of type [SAIS-ST](#)) on the SA Stop Initiation system event on the non-billed budget SA type to automatically finalize and stop the SA (i.e. transition it to stopped status). If you don't use a SA Stop Initiation algorithm, the non-billed budget is stopped the next time the [SA activation background process](#) runs.

To finalize a pending stop SA, the system first calls the stop SA algorithm plugged-in on the SA Stop system event on the SA type. The stop SA algorithm type ([SAST-NB](#)) supplied with the base package:

- Distributes any credit on the non-billed budget to the covered SAs (using the method described in [Distributing Non-billed Budget Credit](#))
- Distributes any excess credit remaining on the non-billed budget using the [overpayment distribution](#) algorithm for the account's customer class and the overpayment transfer adjustment type (specified as a parameter to the algorithm)
- Creates a trigger to cause the account to be reviewed by the account debt monitor
- Creates a customer contact (if the customer contact class and customer contact type parameters are populated)

WARNING:

If you do not plug in an SA stop algorithm that transfers the credit balance from the non-billed budget to its covered SAs (or an excess credit SA), the stopped non-billed budget may have a credit balance. You must then manually distribute this credit.

After the SA stop algorithm is finished, the SA stop processing performs the following steps if the SA type has a special role of non-billed budget:

- If the non-billed budget is monitored, create adjustments to synchronize the current and payoff balance of covered SAs using the **Adj. Type (Synch Current)** adjustment type from the covered SAs' SA types
- Remove the covered SAs from the non-billed budget
- Remove all the scheduled payments from the non-billed budget
- Create an adjustment to synchronize current and payoff on the non-billed budget SA using the **Adj. Type (Synch Current)** adjustment type from the non-billed budget's SA type

NOTE:

Synchronizing current and payoff effectively sets the current amount to zero on the non-billed budget SA, as the payoff amount should have been reduced to zero by the distribution and overpayment processing in the algorithm for SA Stop.

FASTPATH:

Refer to [The Lifecycle Of A Service Agreement](#) for more information about how a pending stop SA is stopped and closed.

Automatic Payment and Non-billed Budgets

If a customer wants to pay their non-billed budget scheduled payments automatically, the account must be set up for automatic payment. In addition, the non-billed budget must indicate that automatic payment is being used.

FASTPATH:

Refer to [How To Set Up Automatic Payment For A Non-billed Budget](#) for more information.

When this is done, a background process referred to as NBBAPAY creates automatic payments on the scheduled payment date by calling the automatic payment creation algorithm plugged in on the installation record.

NOTE:

You must ensure that your auto pay creation algorithm supports non-billed budget scheduled payments. The [APAY-CREATE](#) algorithm type supplied with the base package supports non-billed budget scheduled payments.

Credit and Collections and Non-billed Budgets

Unless the non-billed budget is [unmonitored](#), the [account debt monitor](#) (ADM) monitors a non-billed budget SA's current amount just as it does for any other SA. The [scheduled payment algorithm](#) creates a trigger to ensure that the account debt monitor reviews the account the next time it runs. The review date on the trigger record is set to the process business date.

A separate [debt class](#) is needed for non-billed budget SA types, thus allowing you to define collection class controls, debt criteria and collection process templates specifically for non-billed budgets. The debt criteria should be set up to trigger a collection process when the arrears amount exceeds \$0.01 for more than n payment periods plus the number of grace days that you want to allow.

The collection process template can perform any of the events in standard collection processes, such as sending letters to customers and creating severance processes. At a minimum, the collection process template should be set up to start a severance process for all service agreements in the debt class. (Since the debt class is specifically for non-billed budgets, the non-billed budget is the only SA that will be subject to a severance process.)

The non-billed budget severance process template should include the following event types:

- Populate a characteristic on the SA to indicate that the SA is broken. The base package comes supplied with a severance event algorithm type ([SVEV-NB](#)) that sets an SA characteristic to indicate that it was "severed".
- [Expire Severance Agreement](#) to move the SA to the pending stop state.

When the system subsequently stops the non-billed budget, the system removes the covered SAs from the non-billed budget and synchronizes their current balances with their payoff balances. Since the SAs have current balances again, they are subject to the account debt monitor, which can start subsequent collection processes for any of the SAs that meet the debt criteria for their debt class.

FASTPATH:

Refer to [Stopping Non-billed Budgets](#) for a complete description of the events that occur when a non-billed budget is stopped.

Customers can catch up on their payments and avoid having their non-billed budget broken as long as their current balance doesn't violate the debt criteria for the non-billed budget's collection process.

Credit and Collections and Unmonitored Non-billed Budgets

If the **Non-billed Budget Monitor** flag on the non-billed budget's SA type is set to Unmonitored, the [NBB Scheduled Payment Processing](#) background process does not create a trigger for the account debt monitor. Additionally, the non-billed budget's current amount is always equal to zero, so it never violates any debt collection criteria. We recommend using a debt class that has the **Eligible for Collection** flag turned off, such as the N/A debt class.

For unmonitored non-billed budgets, the current balance is kept on the covered SAs so they are subject to the account debt monitor and any debt criteria for their SA types' debt classes. For more information, refer to [Financial Transactions for Unmonitored Non-billed Budgets](#).

Non-billed Budget Status

A non-billed budget SA's status is just like any other SA's status. In addition, you can use a characteristic to keep an explicit status relevant to the non-billed budget:

- A base package severance event algorithm type ([SVEV-NB](#)) creates a characteristic value to indicate if a non-billed budget is "broken" (i.e. stopped as a result of a severance process).
- A base package break non-billed budget algorithm type ([NBBR-BRK](#)) creates a characteristic value to indicate if a non-billed budget is "canceled" (i.e. manually stopped by a user).

An active non-billed budget implicitly has a "kept" status (i.e. all scheduled payments have been made).

Alerts For Non-billed Budgets

The system provides [alerts](#) to highlight the existence of non-billed budgets. These alerts are important to assist the customer service representatives:

- An alert is displayed if the account has a non-billed budget that is not stopped (e.g., pending start, active or pending stop).
- When a user denies a non-billed budget (for whatever reason), the user should create a [customer contact](#) with a given [customer contact type](#). This type of alert prevents the customer from shopping around. An existing alert algorithm type ([CC BY TYPCL](#)) can highlight these customer contacts.
- For customers who are permanently forbidden from having a non-billed budget, the user should put a permanent [alert on the account](#).
- Use an algorithm to highlight cancelled or severed non-billed budgets with an entry in the alert zone. The algorithm type to do this is not provided. Use [PP BY STATUS](#) and [CCAL-WF](#) as examples of how to create this type of algorithm.

FASTPATH:

For more information about introducing alert conditions on Control Central, refer to [Installation Options - Algorithms](#).

Non-billed Budget Recommendation Rule

Recommendation rules are used to recommend scheduled payments for non-billed budgets. For information about designing recommendation rules, refer to [Designing Recommendation Rules](#).

To define recommendation rules, navigate to **Admin > Financial > NBB Recommendation Rule > Add**.

Description of Page

Enter an easily recognizable **Recommendation Rule** code and **Description** for each recommendation rule.

Specify the **Average Daily Amount Algorithm** used to calculate the average daily amount for this recommendation rule.

Specify the **Payment Schedule Algorithm Type** used to create the recommended payment scheduled for non-billed budgets that use this recommendation rule. The Payment Schedule Algorithm Type cannot be modified if a non-billed budget SA that is not stopped or cancelled is using this recommendation rule.

Payment Schedule Parameters enables you to define collections of default parameter values for the payment schedule algorithm type that are effective dated. For each collection:

- **Effective Date** defines the date on which the collection of parameter values becomes effective.
- **NBB Rule Parameter Value** specifies the default value of each parameter supplied to the algorithm. Note that the [payment schedule algorithm type](#) controls the number and type of parameters.
- **Override Flag** indicates whether the user can override the default value for the parameter.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_NB_RULE](#).

Setting Up The System To Enable Non-billed Budgets

The above topics provided background information about how non-billed budgets are supported in the system. The topics in this section describe how to set up the system to enable non-billed budget functionality.

NOTE:

Example Setup. This section describes typical non-billed budget configurations. Your set up and configuration may differ depending on your business needs. This section is provided for guidance only. Read the descriptions of non-billed budget functionality above to understand the implications of the described setup.

NBB Distribution Codes

You must set up a Non-billed Budget Clearance [distribution code](#) that is used on the non-billed budget SA type to credit non-billed payments.

NBB Adjustment Types

NOTE:

Non-billed Budget Financial Transaction Algorithms. The non-billed budget adjustment types use the standard FT algorithm types that are provided with the base package. If you have not yet defined algorithms for these types in your system, do so before creating the non-billed budget adjustment types.

The following adjustment types are needed:

- **Add SA To Non-billed Budget.** This adjustment type should reference an FT algorithm of Payoff Amt = 0 / Current Amt = Adj Amount (no GL). Because the adjustment affects the current balance only, there is no entry in the GL. This adjustment type is referenced on **Adjustment Type (Current=0)** on SA types that are eligible for non-billed budgets. Note that this adjustment type is never used if the SA is added to an unmonitored non-billed budget.

- Bill Complete for Monitored Non-billed Budget. This adjustment type should reference a distribution code used for balance transfer clearing. It should reference a FT algorithm of $\text{Payoff Amt} = \text{Adj} / \text{Current Amt} = 0$. This adjustment type is referenced on **Adjustment Type (Xfer)** on SA types that are monitored non-billed budgets.
- Synch Balance for Non-billed Budget. This adjustment type should be set up for **Sync. Current Amount** and should reference a FT algorithm of $\text{Payoff Amt} = 0 / \text{Current Amt} = \text{Adj Amount (no GL)}$. This adjustment type is referenced on the **Adj. Type (Synch Current)** on non-billed budget SA types. It is used when a non-billed budget SA is stopped to synch the current balance with the payoff balance.
- Scheduled Payment. This adjustment type should reference an FT algorithm of $\text{Payoff Amt} = 0 / \text{Current Amt} = \text{Adj Amount (no GL)}$. Because the adjustment affects the current balance only, there is no entry in the GL. This adjustment type is referenced on the non-billed budget process scheduled payment algorithm.
- Overpayment Transfer. This adjustment type should reference a transfer distribution code and a FT algorithm of $\text{Payoff Amt} = \text{Adj} / \text{Current Amt} = \text{Adj}$. This adjustment is referenced on the non-billed budget process scheduled payment algorithm and the stop non-billed budget algorithm. Note that if there is already a transfer adjustment type created in your system, you do not need to create a new one.

If you are setting up an unmonitored non-billed budget that maintains a credit balance (as opposed to maintaining the credit balance on an overpayment SA), you need to create an adjustment type for Bill Complete for Unmonitored Non-billed Budget. (Refer to [Transferring Credit from Unmonitored Non-billed Budgets](#) for more information.) The adjustment type should reference a transfer distribution code and a FT algorithm of $\text{Payoff Amt} = \text{Adj} / \text{Current Amt} = \text{Adj}$. This adjustment type is referenced on **Adjustment Type (Xfer)** on SA types that are unmonitored non-billed budgets.

NBB Adjustment Type Profiles

Create an adjustment type profile for non-billed budgets that references the following adjustment types:

- Bill complete for monitored non-billed budget
- Bill complete for unmonitored non-billed budget (if used)
- Synch balance for non-billed budget
- Scheduled payment
- Overpayment transfer

Create an adjustment type profile for eligible SAs that references the following adjustment types:

- Add SAs to monitored non-billed budgets
- Bill complete for monitored non-billed budget
- Bill complete for unmonitored non-billed budget (if used)

NOTE:

Bill Complete Adjustment Types. Because the bill complete adjustment types transfer amounts between two SAs, they must be in profiles for both non-billed budget and eligible SA types.

Overpayment Transfer Adjustment Type. The overpayment transfer adjustment type created above is used to transfer funds from an excess credit SA to a non-billed budget when the scheduled payment is processed. It is also used to transfer excess funds from a non-billed budget that is being closed to an excess credit SA. The transfer adjustment should therefore be added to an adjustment type profile that is referenced on the excess credit SA type.

NBB Characteristic Types

Create a non-billed budget status characteristic type that specifies Service Agreement as its characteristic entity. The characteristic type should include the following predefined values:

- Non-billed Budget Canceled
- Non-billed Budget Severed

NBB Customer Contact Class And Types

Create a non-billed budget customer contact class. The contact class may include the following customer contact types:

- Non-billed Budget Activate
- Non-billed Budget Renewal
- Non-billed Budget Stop

NOTE:

Customer Contact Letters. If you want to send letters to your customers when a contact of any of these types is created, you must create an appropriate [letter template](#) and attach it to the contact type.

NBB Algorithms

You must define the following [algorithms](#):

- On [NBB recommendation rule](#), the Non-billed Budget Daily Amount Calculation. Refer to the [NBDA-DA](#) algorithm type for more information about the base package algorithm.
- On [SA type](#):
 - Non-billed Budget Process Scheduled Payment. This algorithm has parameters that must reference the Scheduled Payment and Overpayment Transfer adjustment types defined above. Another parameter references an overpayment SA type (that you may need to create if there is not already one in your system). Refer to the [NBPA-PS](#) algorithm type for more information about the base package algorithm.
 - Non-billed Budget SA Renewal. Refer to the [SARN-NB](#) algorithm type for more information about the base package algorithm.
 - Break Non-billed Budget SA. Refer to the [NBBR-BRK](#) algorithm type for more information about the base package algorithm.
 - SA Activation - automatically activate SA (if you want to automatically activated non-billed budgets when they are created). Refer to the [SACR-AT](#) algorithm type for more information about the base package algorithm. Note that this same algorithm may be used on many SA types.
 - SA Activation - create customer contact (if you want the system to create a customer contact when NBB SAs are activated). Refer to the [SAAT-CC](#) algorithm type for more information about the base package algorithm.
 - SA Stop - automatically stop SA (if you want to automatically transition non-billed budgets from pending stop to stop when their stop is initiated). Refer to the [SAIS-ST](#) algorithm type for more information about the base package algorithm. Note that this same algorithm may be used on many SA types.
 - SA Stop - Stop Non-billed Budget. Refer to the [SAST-NB](#) algorithm type for more information about the base package algorithm.
 - Bill Completion - Non-billed Budget Credit Transfer. Refer to the [BCMP-NB](#) algorithm type for more information about the base package algorithm.

- On [bill segment type](#), the Bill FT Algorithm (for all SAs that are eligible for NBB). Refer to the [BSBF-BA](#) algorithm type for more information about the base package algorithm. Note this is the standard bill FT algorithm type used for common bill transactions. It supports bill FT for non-billed budgets and other SA types. You only need to create it if it doesn't already exist in your system.
- On [severance event type](#), an algorithm for Non-billed Budget Severance. Refer to the [SVEV-NB](#) algorithm type for more information about the base package algorithm.

NOTE:

Payment Schedule Algorithm Types. For non-billed budget payment schedule algorithm types, you need to define the algorithm types (if you add your own algorithm types). You do not need to define algorithms because the parameter values for the algorithm are defaulted on the recommendation rule and stored with the non-billed budget SA. (Normally the algorithm holds the parameter values.) Refer to the [NBPS-MON](#) algorithm type for more information about the base package payment schedule algorithm.

NBB Debt Class And Collection Process

Set up a separate debt class, collection class control, collection process template and severance process template for non-billed budgets according to the information in [Credit and Collections and Non-billed Budgets](#).

SA Types for SAs Covered by NBBs

You must modify the SA type for any SAs that you want to allow to be covered by a non-billed budget.

- Verify that the specified **Bill Segment Type** (on the Billing page) references a financial transaction algorithm to set the current amount to zero for monitored non-billed budgets. The FT creation algorithm type [BSBF-BA](#) (i.e., $\text{Payoff Amt} = \text{Bill Amt} / \text{Current Amt} = \text{Amt Due}$) supplied with the base package sets the current amount to zero for SAs that are covered by monitored non-billed budgets, though this is not evident by the name. (For SAs that are not covered by a non-billed budget, the current amount is equal to the amount due or the recurring charge.) Refer to [Billing For SAs Covered By The Non-billed Budget](#) for more information.
- Set the **Eligible for Non Billed Budget** flag (on the Billing page) to Eligible for Non-billed Budget .
- Populate the **Adjustment Type (Current = 0)** (on the Detail page) to indicate the adjustment to be used to zero out the current amount on the covered SAs when the non-billed budget SA is activated. Use the [Add SA To Non-billed Budget adjustment](#) created above. This adjustment type is only called for SAs that are covered by a monitored non-billed budget.
- Reference an **Adjustment Type Profile** (on the Adjustment Profile page) that includes the [Add SA To Non-billed Budget adjustment](#) type referenced in the **Adjustment Type (Current = 0)** field above.
- If not already specified (for write off), an **Adj. Type (Synch Current)** (on the Main page) is also required. It is used to synchronize (make equal) the current amount with the payoff amount when the SA is removed from (i.e. no longer covered by) a non-billed budget.

NBB Recommendation Rules

Set up any [NBB Recommendation Rules](#) that you want to be available for your non-billed budget SAs. Use any Non-billed Budget Daily Amount Calculation algorithms defined above. Also use the Non-billed Budget Monthly Payment Schedule ([NBPS-MON](#)) algorithm types and specify the default parameters.

Non-billed Budget SA Types

You must set up a [SA types](#) for your non-billed budget service agreements. You may need multiple non-billed budget SA types if you have different business rules for different types of non-billed budgets, for example, if you have both monitored and unmonitored non-billed budgets or if you support non-billed budgets with different renewal requirements. The following points provide guidelines for creating a non-billed budget SA type.

SA Type - Main (NBB)

Service Type should reference something like "Miscellaneous Service".

Distribution Code should be the one you set up to book credits for non-billed budgets.

Revenue Class should be set to N/A. (Revenue classes are not applicable because non-billed budgets do not apply a rate and revenue classes are only relevant for SA types that use a rate.)

The **Payment Segment Type** should reference the Normal Payment.

Do Not Overpay should be on. Any excess payments should go to the overpayment SA, not the non-billed budget SA.

Late Payment Charge is not applicable and should not be turned on because non-billed budgets are not billed.

Adj. Type (Synch Current) should reference the Synch Balance for Non-billed Budget adjustment type created above.

SA Type - Detail (NBB)

- **Special Role** is Non-billed Budget.
- **Adjustment Type (Xfer)** must be populated to indicate the adjustment to be used for transferring accumulated credit from the non-billed budget SA to the covered SAs. This field is not used for unmonitored non-billed budgets.
- **Renewal** may be optional, not allowed, or required depending on your business processes.
- If Renewal is required, specify the **Days Before Expiration for Renewal**.
- **Non-billed Budget Monitoring** must indicate whether the non-billed budget is monitored by the account debt monitor.

SA Type - Billing (NBB)

Non-billed budget SAs do not get billed, so the **Eligible for Billing** flag should be off.

Additionally, the **Characteristic Premise Required** should not be checked for non-billed budgets.

SA Type - Rate (NBB)

Non-billed budget SAs do not use rates, so the **Rate Required** flag should be off.

SA Type - SP Type (NBB)

Non-billed budget SAs do not have service points, so the **Service Points Required** flag should be off.

SA Type - Adjustment Profile (NBB)

Adjustment Type Profile should reference the [adjustment type profile](#) for non-billed budgets (set up above).

SA Type - Credit and Collections (NBB)

The credit and collections information should reference a **Severance Process Template** that calls the Non-billed Budget Severance algorithm created above and expires the non-billed budget SA. The **Debt Class** should reference an appropriate value consistent with your credit and collections rules. Typically, monitored non-billed budgets should be in their own debt class. Unmonitored non-billed budgets should have a **Debt Class** that is not Eligible for Collection.

The **Write Off Debt Class** is not applicable because the non-billed budget contains no debt to be written off. Reference a debt class such as N/A. Refer to [Defining Credit & Collections Options](#) for more information.

SA Type - Algorithms (NBB)

The SA type [algorithms](#) defined above must be set up:

- The Non-billed Budget Credit Transfer algorithm created above should be specified for the Bill Completion system event (not used on unmonitored non-billed budgets).
- The Break Non-billed Budget SA algorithm created above should be specified for the Break NBB SA system event.
- The Non-billed Budget Process Scheduled Payment algorithm created above should be specified for the Process NBB Scheduled Payment system event (not used on unmonitored non-billed budgets).
- The Non-billed Budget SA Renewal algorithm created above should be specified for the SA Renewal system event.
- If you created the Automatic SA Activation algorithm above, it should be specified for the SA Creation system event.
- The Non-billed Budget SA Activation algorithm created above should be specified for the SA Activation system event.
- If you created the Automatic SA Stop algorithm above, it should be specified for the SA Stop Initiation system event.
- The Stop Non-billed Budget algorithm created above should be specified for the SA Stop system event.

SA Type - NBB Recommendation Rule (NBB)

Add the recommendation rules defined above that are valid for SAs of this type. Also, indicate which recommendation rule should be used as the default.

NBB Background Processes

Ensure that the following background processes are scheduled:

- Non-billed Budget Scheduled Payment Processing ([NBBPS](#))
- Non-billed Budget Scheduled Payment Automatic Payment Create ([NBBAPAY](#))
- Service Agreement Renewal ([SARENEW](#))

- Stop Expired Service Agreements ([SAEXPIRE](#))

Defining Quotation Options

This section describes tables that must be set up before quotations can be created.

FASTPATH:

For more information, refer to [The Big Picture of Quotations](#).

Setting Up SA Types For Quotes

The topics in this section describe additional setup responsibilities required on SA types that can have proposal service agreements.

NOTE:

Assumption. We have assumed that you've already designed your SA types for the services that you sell. If you haven't done this, refer to [Defining Service Agreement Types](#).

Enabling The Automatic Generation Of Billing Scenarios

As described under [Proposal SAs Contain Billing Scenarios And Template Consumption](#), a proposal SA requires billing scenarios before a quote detail can be generated. The system will automatically create billing scenarios when a proposal SA is created if you plug-in the appropriate Proposal SA Creation algorithm on each such [SA type](#). Refer to [PSAC-CBS](#) for an example algorithm (note, this algorithm can be used to create billing scenarios for both interval and non-interval service agreements).

Enabling The Generation Of Simulated Bill Segments

As described under [Creating Quotes And Quote Details](#), in order for the system to generate simulated bill segments for a proposal SA, you must plug-in a Proposal SA Bill Segment Generation algorithm on the proposal SA's [SA type](#). Refer to [CBSP-AR](#) for an example algorithm that creates a simulated bill segments for each billing scenario linked to the proposal SA by calling rate application.

NOTE:

Interval pricing service agreements. If you have proposal SAs that require the derivation of interval profiles from other interval profiles, you must also plug-in another Proposal SA Bill Segment Generation on the interval [SA type](#). Refer to [CBSP-IDDRV](#) for an example algorithm that performs derivation. When you plug-in this type of algorithm, don't forget to use a sequence number less than the one that generates the simulated bill segments (see above); otherwise, the derived interval profiles won't exist when rate application is called.

Enabling The Creation Of A Real SA When A Quote Detail Is Accepted

As described under [Accepting / Declining Quote Details](#), in order for the system to create a real SA when a proposal SA is accepted, you must plug-in a Proposal SA Acceptance algorithm on the proposal SA's [SA type](#). Refer to [PSAA-PS](#) for an example algorithm that creates a real SA by copying the proposal SA.

NOTE:

Interval pricing service agreements. If the service agreement has SA-specific interval profiles, time-of-use maps, or contract options, it's important that you setup your Proposal SA Acceptance algorithm to create fresh interval profiles, time-of-use maps and contract options when the real SA is created (there's a parameter on [PSAA-PS](#) that will do this). Refer to [Proposal SAs And Interval Consumption](#) for more information.

Setting Up Quote Route Types

Quote route types control how quotes are [routed](#) to customers and prospects. To define a quote route type, open **Admin > Sales & Marketing > Quote Route Type**.

FASTPATH:

Refer to [Printing Quotes](#) for more information about how quotes are routed to customers and prospects.

Description of Page

Enter a unique **Quote Route Type** and **Description** for every quote route type.

Quote Routing Method controls the type of information that may be defined when the respective **Quote Route Type** is selected on [Account - Person Information](#). The following options are available:

- Postal: Use this method if the routing is via the postal service.
 - Fax: Use this method if the routing is via fax.
 - Email: Use this method if the routing is via email.
-

NOTE:

The values for **Quote Routing Method** are customizable using the [Lookup](#) table. This field name is QTE_RTG_METH_FLG .

- The next two fields control how quotes that are routed using this method are [printed](#) (both in batch and online).
 - Use **Batch Control** to define the process that creates the flat file that is passed to your quote printing software. Refer to [Technical Implementation of Printing Quotes In Batch](#) for more information about these processes.
 - Use **Extract Algorithm** to define the plug-in that constructs the "flat file records" that contain the information merged onto quotes routed using this method. Refer to [Printing Quotes](#) for more information.
-

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_QTE_RTE_TYPE](#).

Setting Up Terms and Conditions

As described under [Legal Terms and Conditions May Be Specified On SAs](#), your [SA type start options](#) can reference terms and conditions (T&C's) that should be defaulted onto new service agreements (both real and proposal). Each T&C is identified with a terms and condition code. To define a terms and conditions code, open **Admin > Terms and Conditions > Add**.

NOTE:

T&C print order. The value of the T&C code controls the order in which the T&C appears on the printed quote. This means you should assign these codes in some type of structured format (e.g., 01...) if you would like them to appear in a certain order.

Description of Page

Enter a unique **Terms and Condition** code and **Description**. Use **Text** to describe the exact terms.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_TC](#).

Setting Up Decline Reasons

A proposal SA decline reason must be supplied when a proposal SA is declined. Open **Admin > Sales & Marketing > Proposal SA Decline Reason** to define your reason codes.

Description of Page

Enter an easily recognizable **Decline Reason** and **Description** for each proposal SA decline reason.

Where Used

Decline reasons are used when a [quote detail is declined](#).

Setting Up Customer Classes For Quotes

An optional plug-in spot exists on [customer class](#) where you can introduce additional logic to be executed when a quote is completed for an account that belongs to this customer class. The base package comes supplied with a sample algorithm that creates a workflow process when a quote is completed (refer to the algorithm [QTEC-WP](#) for more information).

Defining Case Management Options

Case management functionality is a highly configurable tool your organization can use to manage many situations, including (but certainly not limited to) the following:

- a high-bill complaint,
- a bankruptcy,
- an inspection of a premise,
- a customer's request for literature,
- an application for new service,
- a contractor's request to extend a line,
- a customer's rejection of a quote,
- a customer's request to change service providers on a future date,
- the processing of a market message in a deregulated environment,
- ... (the list is only limited by your time and imagination)

Obviously the steps involved in the resolution of the above cases are very different. The topics in this section describe how to configure the system to manage your cases as per your organization's desires.

NOTE:

Separate module. The Case Management functionality is associated with a separate module. If the Case Management module is not applicable to your business you may turn it off. Refer to [Turn Off A Function Module](#) for more information.

FASTPATH:

Refer to [Case Management](#) for a description of how end-users use cases.

The Big Picture Of Cases

The topics in this section provide background information about how to configure the system to support your case management requirements.

Case Type Controls Everything

Whenever a user creates a case, they must specify the type of case (e.g., high-bill complaint, literature request, etc.). The case type controls how the case is handled.

Case types hold the business rules that control cases. Since these business rules can sometimes be quite complicated, setting up case types requires planning and foresight. The topics in this section describe the type of business rules that can be configured on your case types.

Person / Account / Premise Applicability

Some types of cases may be person-oriented, others may be premise-oriented, and still others may be account-oriented. For example:

- Cases used to keep track of a literature request would reference the person who requested the literature.
- Cases used to keep track of the inspection of a property would reference the premise being inspected.
- Cases used to keep track of a high-bill complaint would reference the account associated with this bill(s) being disputed.

When you set up a case type, you define if its cases must reference a person, account, and/or premise. Note, any combination of these objects is permitted on a case.

Contact Information Applicability

When a case is created, you may want to keep track of how to contact its originator. For example, you may want to record the originator's email address or phone number. When you set up a case type, you define if contact information is required, optional or not allowed on its cases.

Business Object Association

A case type may reference a Business Object, which serves as a link between cases of that type and the options that are associated with the business object.

Additional Information

Some of your cases may require additional information (in the form of [characteristics](#)). For example, a high-bill complaint may require at least one bill. When you set up a case type, you can define the additional fields that are required. In addition, you can define default values for these fields.

The case functionality also allows you to require characteristics when a case enters a given state. Refer to [Required Fields Before A Case Enters A State](#) for the details.

NOTE:

Requiring supporting documents. Because any [type of characteristic](#) can be referenced on a case, you can require references to supporting documents by requiring a file location characteristic.

Access Rights

You can take advantage of the system's [security](#) to restrict cases of a given type to certain users. For example, you can restrict high-bill complaints to specific user groups.

The following points describe how to implement this type of security:

- Create an [application service](#) for each type of case you need to secure
- Define the access modes Add, Inquire and Change for each application service
- Define the applicable application service on each case type
- Link the appropriate [user groups](#) to each application service
 - For user groups that are allowed to add cases of a given type, define Add as a valid access mode.
 - For user groups that are allowed to view cases of a given type, define Inquire as a valid access mode
 - For user groups that are allowed to change cases of a given type, define Change as a valid access mode

If you restrict access to a case type's cases, you can further restrict which users can work on cases given the status of the case. Refer to [Which Users Can Transition A Case](#) for more information.

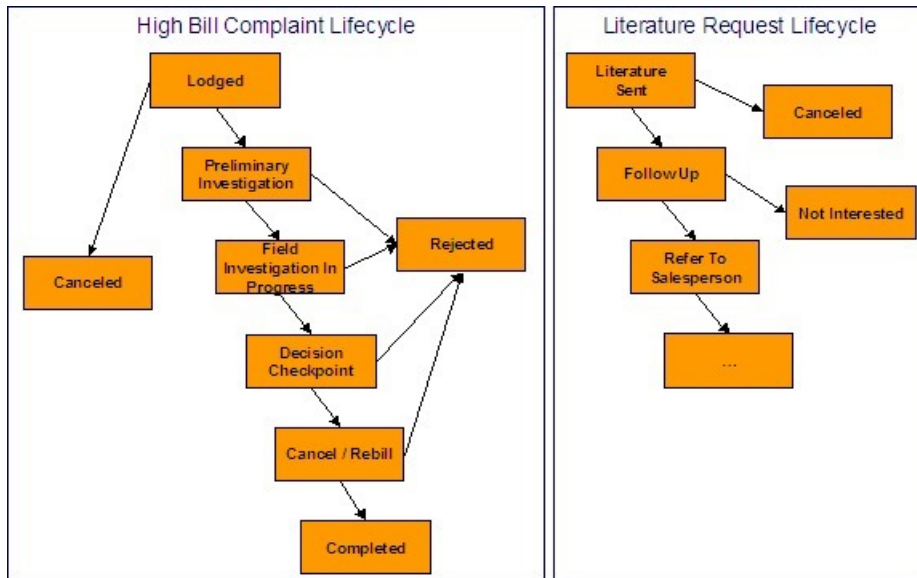
NOTE:

Restricting access to cases is optional. If you don't specify an application service on a case type, all users (who have access to the case transaction) may access its cases.

Lifecycle

Many objects in the system have predefined lifecycles whose rules are governed by the base-package and cannot be changed. For example, a service agreement starts out in the Pending Start state and eventually becomes Closed when it's been final billed (and paid). You can't change the system to allow a service agreement to start its life in the Closed state.

The lifecycle of cases is not governed by the base package. Rather, you define the lifecycle of your cases when you set up their case types. Examine the following diagrams; the one on the left shows the potential lifecycle of a case that manages a high-bill complaint, the one on the right shows the potential lifecycle of a case that manages a customer's literature request.



Potential Lifecycles Of Two Types Of Cases

NOTE:

Just examples. The above lifecycles are just examples. When you set up your case types, you must define the valid states for your case type.

The topics that follow describe important concepts that are illustrated in the above diagrams.

Valid States versus State Transition Rules

The orange boxes in the above diagram show the potential valid states a given case can have. The lines between the boxes indicate the state transition rules. These rules govern the states a case can move to while it's in a given state. For example, the above diagram indicates a high bill complaint that's in the Lodged state can be either Canceled or moved into the Preliminary Investigation state.

When you set up a case type, you define both its valid states and the state transition rules.

Transitory States

You can define a state in a case type as **Transitory** if you do not wish the case to exist in a particular state. For example, let's assume that an algorithm is associated with the Decision Checkpoint state (Enter Processing) that would automatically determine the next state for the case (i.e. Cancel/Rebill or Reject) and also contains logic to transition the case accordingly. In this scenario, you may not ever want the case to exist in the Decision Checkpoint state, so that a user won't ever see a high bill complaint in that state. If the other states were marked as non-transitory, and an error were to occur during the transition from Decision Checkpoint to Cancel/Rebill, the case would roll back any changes to data made in the Cancel/Rebill (Enter Processing) state along with the changes made in the Decision Checkpoint state, and would end up in the Field Investigation In Progress state - the last non-transitory state prior to Decision Checkpoint.

One Initial State and Multiple Final States

When you set up a case type's states, you must pick one as the initial state. The initial state is the state assigned to new cases of a given type. For example, high-bill complaint cases have an initial state of Lodged, whereas literature request cases have an initial state of Literature Sent.

You must also define which statuses are considered to be "final". When a case enters a "final" state, it is complete and no further action is necessary. You might want to think of the "final" states as the potential outcomes of a case. For example, a high-bill complaint has potential outcomes of Completed, Rejected, and Canceled.

The "final" states are used by the system to differentiate between open and closed cases. For example, an alert highlights when the person / account / premise in context has open cases (this alert only exists if you've plugged-in the appropriate installation [alert](#)).

Allowing A Case To Be Reopened

You can set up your state transition rules to allow a case to be reopened (i.e., to be moved from a final state to a non-final state). Neither of the above examples allows this, but it is possible if you configure a case type accordingly.

Make Sure To Have A Canceled State

The system does not allow you to delete a case. Therefore, if you want to support logical deletion, you should have a status of Canceled early in a case type's lifecycle. Doing this allows a user to cancel (i.e., logically delete) a case.

NOTE:

Cancel reason. You might want to consider setting up your case types to require a cancel reason (in the form of a [predefined value characteristic](#)) when a user cancels a case. Refer to [Required Fields Before A Case Enters A State](#) for more information.

Buttons Are Used To Transition A Case From Status To Status

When a case is displayed on [Case - Main](#), a separate button is shown for each state into which the case can be transitioned. For example, a high-bill complaint case that is in the Lodged state would show two buttons: **Start Investigation** and **Cancel**. If the user presses the **Start Investigation** button, the case is transitioned to the Preliminary Investigation state. If the user presses the **Cancel** button, the case is moved to the Canceled state.

You may define the text displayed on the button differently for each state transition. This allows the action description to be varied according to the previous status. For example, the button to transition from New to Active may be labeled **Activate**, but the button to change from Closed to Active may be labeled **Reactivate**.

Refer to [Which Users Can Transition A Case](#) for instructions describing how to restrict users to specific actions.

State Transitions Are Audited

The system maintains an audit trail whenever a case transitions from one state to another. This audit is shown in the case's [log](#).

Reports and Analytics Highlight Productivity

When you set up a case type's lifecycle, keep in mind that several reports and analytics highlight how long it took cases to transition into a state. For example, you can use a report to see how long it took high-bill complaints to be completed (or initially actioned or ...). Refer to the [Reports](#) chapter for the details of case reports.

Status-Specific Business Rules

As described in [Lifecycle](#), when you set up a case type, you define the possible states its cases can pass through. The following topics describe business rules that can be configured for each state.

A Script That Helps A User Work Through A Case

You can define a [Business Process Assistant script](#) that helps a user work a case while it's in a given state. For example, when you set up the Preliminary Investigation state for the high-bill complaint case type, you can define a script. A user can then easily launch this script to help them work through a case in this state.

Please keep the following in mind when you're designing how to integrate BPA scripts with your cases:

- You can have a different script for each state. For example, you could develop a script to help a user work on a case while it's in the Preliminary Investigation state and a different script to help them work in a case while it's in the Decision Checkpoint state.
- Rather than make a user launch a script by pressing a hyperlink on the [case page](#), you can have the system automatically launch the script while the case is in a given state. Refer to [Script Launching Option](#) for more information.
- You can also have the system automatically launch a script when a user selects a To Do entry. Refer to [Launching Scripts When To Do Entries Are Selected](#) for more information.

FASTPATH:

Refer to [Scripts and Cases](#) for more information about how to streamline your case processing with scripts.

Required Fields Before A Case Enters A State

You can define additional fields (i.e., characteristics) that are required before a case can enter a given state. For example,

- You can indicate a high-bill complaint must reference at least one bill before it enters the Preliminary Investigation state.
- You can indicate a case must reference a cancel reason before it enters the Canceled state.

You do this by indicating that [characteristics](#) (that were optional when the case was added) are required when a case enters a given state.

Validation Before A Case Enters A State

You can define validation that executes before a case can enter a given state. For example, you can indicate the case must have been assigned a responsible user before it can enter the Preliminary Investigation state. This validation logic is held in algorithms that are plugged in on the case type and therefore you can define any type of validation.

Additional Processing When Entering A State

You can define additional processing that should happen when a case enters a given state. For example, you can have a [letter](#) created when a high-bill complaint case is Rejected. Similarly, you can have a [To Do entry](#) created when a high bill complaint enters the Preliminary Investigation state. This additional processing is held in algorithms that are plugged in on the case type and therefore you can define any type of additional processing.

You can also incorporate state transition logic within routines that are executed when a case enters a state, so that you do not need to rely upon CASETRAN to transition your cases. For example, when the state entry routines of the Preliminary Investigation status for a high-bill complaint are executed, they may be designed to transition the case into either the Rejected or Field Investigation In Progress state without waiting. Note that your Exit Validation and Exit Processing logic, if configured for the case state, will still be executed as part of the state transition. Auto-Transition logic for this state will be ignored during this transition.

Validation Before A Case Exits A State

You can define validation that executes before a case can exit a given state. For example, you might want to check the account's balance is less than a given value before a case can exit a given state. This validation logic is held in algorithms that are plugged in on the case type and therefore you can define any type of validation.

Additional Processing When Exiting A State

You can define additional processing that should happen when a case exits a given state. For example, you can have a [To Do entry](#) automatically completed when a high bill complaint leaves the Decision Checkpoint state. This additional processing is held in algorithms that are plugged in on the case type and therefore you can define any type of additional processing.

Automatic Transition Rules

You can define rules that automatically transition a case into a different state. For example, you can indicate a literature request should be transitioned to the Follow Up state 1 week after the literature is sent. Similarly, you can indicate a high-bill complaint should transition to the Decision Checkpoint state after the fieldwork is complete. These rules are held in algorithms that are plugged in on the case type and therefore you can define any type of automatic transition rules.

Cases in a state with automatic transition rules are monitored by the [CASETRAN](#) background process. Each time this program runs, the respective automatic transition plug-in is called for each such case and it transitions the case if the condition applies.

NOTE:

When to execute CASETRAN. Because your automatic transition rules will be dependent on your business requirements, you need to think carefully about when you run the [CASETRAN](#) background process. For example, if you have automatic transition rules that transition a case to a new state when a related field activity is completed, you would want to schedule this job to run after field activities are uploaded. If you have rules to transition a case after a customer pays a deposit, you'd want to schedule this job to run after payments are uploaded. Bottom line - your business rules will dictate the frequency of execution.

When the user adds a new case or changes the state of a case manually the system attempts to auto-transition the case to subsequent statuses as necessary. If auto-transition rules apply to the new state (and to subsequent ones) they would be executed right away. In other words, you don't need to wait for the auto-transition background process to be executed. An indication that the case was auto-transitioned online is displayed right below the action buttons section.

NOTE:

Auto-Transition Errors. Online auto-transition is performed recursively committing each successful state transition to the database. It is performed up to 100 times or until an error is encountered during the process. If this happens, auto-transition stops at the last **non-transitory** state into which a successful transition had occurred. Two case log entries will be generated automatically - one containing the message that a transition error has occurred, and a second containing the actual error message. A To Do entry will also be generated automatically upon rollback. The type of this To Do entry will be taken from 1) the Case Transition Exception To Do Type **option** for the **Business Object** associated with the case type, and if this is not populated, 2) the Exception To Do Type indicated on the Case Options Feature Configuration. All of the above error handling is true for both batch and online processing of cases.

NOTE:

Triggering Auto-Transition. If you have a customized process that affects the state of a case and you want the case to be auto-transitioned right away, i.e. not wait for the next scheduled **CASETRAN** background process to execute, you can customize that process to trigger auto-transition for the specific case, or you can put the state transition logic into the routines that execute at state entry time.

Script Launching Option

You can define whether the script associated with a given state is to be automatically launched while the case is in that state. The system supports the following options:

- Launch the script only if no script is currently active.
 - Always launch the script unless this specific script is currently active.
-

WARNING:

With this option, if a script is currently open in the page's BPA script area then it will be automatically closed and the case script will open.

- Do not automatically launch the script.

You do this by plugging-in a Script Launching algorithm for the given state. If no such plug-in is provided the script is not automatically launched.

Which Users Can Transition A Case Into A State

If you have **restricted access** to a case type, you can further restrict which user groups are allowed to transition a case into specific states. For example, you can control which user group can transition a high bill complaint into the Preliminary Investigation state. The following points describe how this is done:

- Define actions on the **application service** defined on the case type. You must define an action for each status that you need to secure.
- Define each status's corresponding action. Note, you only need to link a status to an action if it's secured. Any user with **access** to the case type can perform statuses that aren't linked to actions.
- Define the transition role for each status's valid next status. You can assign valid next statuses to be reachable via system (only), or system and user.
- Define which **user groups** have access to the actions (i.e., statuses). In addition, these user groups should have access to the Change action.

Responsible User Applicability

Some of your cases may require a "responsible user". This is the user who has overall responsibility for the case. When you set up a case type, you define if a responsible user is required, optional or not allowed on its cases.

The following points describe how to set up the system if a responsible user is not required when a case is first created, but is later in its lifecycle:

- Indicate that a responsible user is optional on the case type
- Plug-in either an [exit validation](#) or [entry validation](#) algorithm on one of the case type's states to require a responsible user at some point in a case's lifecycle

NOTE:

Address To Do entries to the responsible user. If you use the [base-package algorithm](#) to create a To Do entry when a case enters a given state, you can indicate that the To Do entry should be addressed to the responsible user on the case.

Scripts and Cases

There are three ways [Business Process Assistant scripts](#) can be used to manage cases:

- You can create a BPA script to help users create a case. For example, a script can help a user create a new high-bill complaint.
- Using a script to create a case can save a user a lot of time (and training efforts). This is because the script can automatically populate many fields on the case based on answers to questions.

Refer to [Initiating Scripts](#) for a description of how end-users initiate scripts.

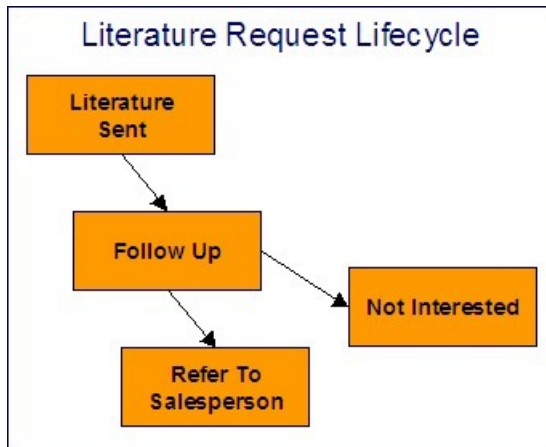
- You can create a script to help users work on a case when it's in a given state. Refer to [A Script That Helps A User Work A Case](#) for more information.
- You can [set up your case types to create To Do entries](#) to notify users when cases exist that require their attention. Users can complete many of these ToDo entries without assistance. However, you can set up the system to automatically launch a script when a user selects a ToDo entry. For example, consider a ToDo entry that highlights a high-bill complaint that requires investigation. You can set up the system to execute a specific script when a user selects this ToDo entry. This script might guide the user through the investigation process (and help them update the case). Refer to [Executing A Script When A To Do Entry Is Selected](#) for more information.

To Do's and Cases

The topics in this section provide background information about how to facilitate case management with [To Do entries](#).

Creating and Completing To Do Entries

You can configure your case types to create and complete [To Do entries](#) when a case enters or exits a state. Let's use the following [lifecycle diagram](#) to illustrate a potential use of To Do's:



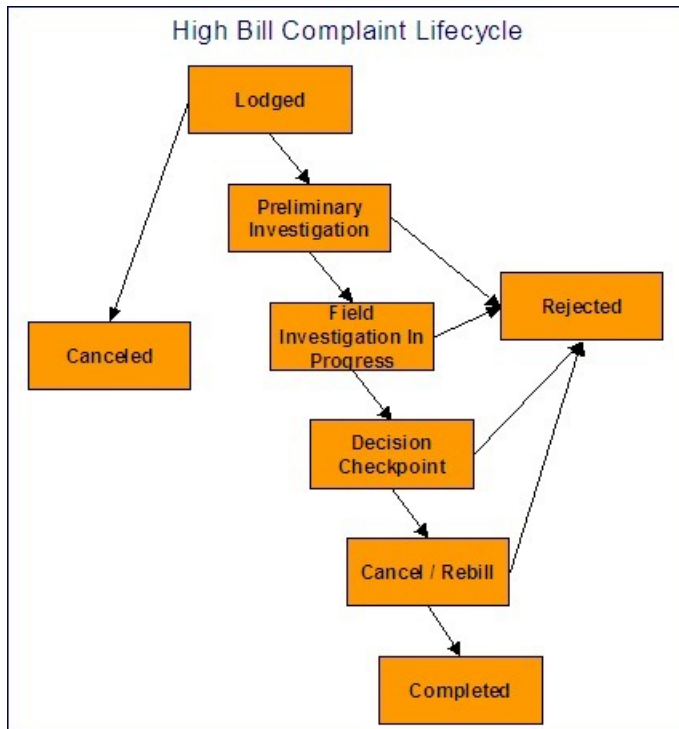
Let's assume the following:

- You want a To Do entry created when a literature request case enters the Follow Up state. You want this To Do automatically completed when the case enters either the Refer To Salesperson or Not Interested states. Note, we refer to this as the "first" To Do entry below.
- You want a different To Do entry created when a case enters the Refer To Salesperson state. You do not want the system to automatically complete this entry (the sales person must manually do this). Note, we refer to this as the "second" To Do entry below.

To implement the above, you would set up a case type as follows:

- Plug-in an [entry processing](#) algorithm on the Follow Up status to create the first To Do entry.
- Plug-in an [exit processing](#) algorithm on the Follow Up status to complete the first To Do entry.
- Plug-in an [entry processing](#) algorithm on the Refer To Salesperson status to create the second To Do entry.

While the case type illustrated above had a single To Do entry "active" at any point in time, you can easily configure a case type to have multiple To Do entries active at any point in time. Let's use the following lifecycle diagram to illustrate this point:



Let's assume the following:

- You want a To Do entry created when a high bill complaint is created and you want it completed when the case reaches the Canceled, Rejected or Approved states. This To Do entry could be used by a supervisor to monitor the number of high-bill complaints being worked. Note, we refer to this as the "first" To Do entry below.
- You want a different To Do entry created when the case enters the Preliminary Investigation state and you want this entry automatically completed when the case leaves this state. Note, we refer to this as the "second" To Do entry below.
- You want a different To Do entry created when the case enters the Decision Checkpoint state and you want this entry automatically completed when the case leaves this state. Note, we refer to this as the "third" To Do entry below.

To implement the above, you would set up the case type as follows:

- Plug-in an [entry processing](#) algorithm on the Lodged status to create the first To Do entry. Plug-in an [entry processing](#) algorithm on the Canceled, Rejected and Completed statuses to complete this entry.
- Plug-in an [entry processing](#) algorithm on the Preliminary Investigation status to create the second To Do entry. Plug-in an [exit processing](#) algorithm on the Preliminary Investigation status to complete this entry. We elected to use an exit processing algorithm because we only have to plug it in on one status. If we'd used an entry processing algorithm, we would need to plug it in on the 2 statuses into which a Preliminary Investigation status can transition.
- Plug-in an [entry processing](#) algorithm on the Decision Checkpoint status to create the third To Do entry. Plug-in an [exit processing](#) algorithm on the Decision Checkpoint status to complete this entry.

Launching Scripts When To Do Entries Are Selected

You can set up your case types to create To Do entries to notify users when cases exist that require their attention. Users can complete many of these To Do entries without assistance. However, you can set up the system to automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a high-bill complaint that requires investigation. You can set up the system to execute a specific script when a user selects this type of To Do entry. This script might guide the user through the investigation process. Refer to [Executing A Script When A To Do Entry Is Selected](#) for more information.

All To Do Entries Are Visible

When a case is displayed on [Case Maintenance](#), the system summarizes the number of To Do entries associated with the case (if you've [set up your To Do types](#) appropriately).

Examples of Case Types

The topics that follow provide examples of case types related to several business processes. Use the information in this section to form an intuitive understanding of case types. After attaining this understanding, you'll be ready to design your own case types.

High Bill Complaint

Some organizations will set up a case to manage a high-bill complaint. The following diagram illustrates how such a case type might look:

Case Type – High Bill Complaint

Person / Account / Premise Applicability	
Field	Applicability
Person	Required
Account	Required
Premise	Optional

Secured

Yes, all actions

Fields To Be Captured		
Field	Required / Optional	Default Value
Bill ID	Optional	**
Cancel reason	Optional	**
Reject reason	Optional	**

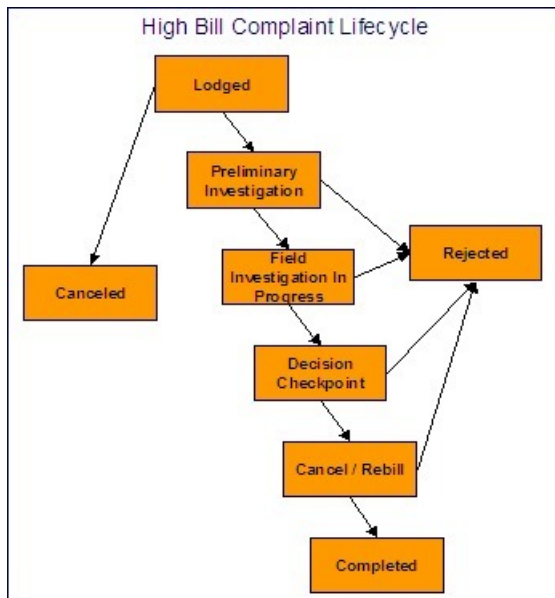
Applicability Rules	
Field	Applicability
Responsible User	Optional
Contact Information	Optional

Note the following about the High Bill Complaint case type:

- Notice that we set up the case type to require a person and an account, but premise is optional. This is because a bill can span multiple premises and knowing the premise isn't so important on cases of this type.
- We need to restrict high-bill complaints to specific user groups. This means we need to [set up a specific application service](#) for this case type (that has a separate "action" for each status).
- Cases of this type have 3 additional fields over their lifetime. Notice the following:

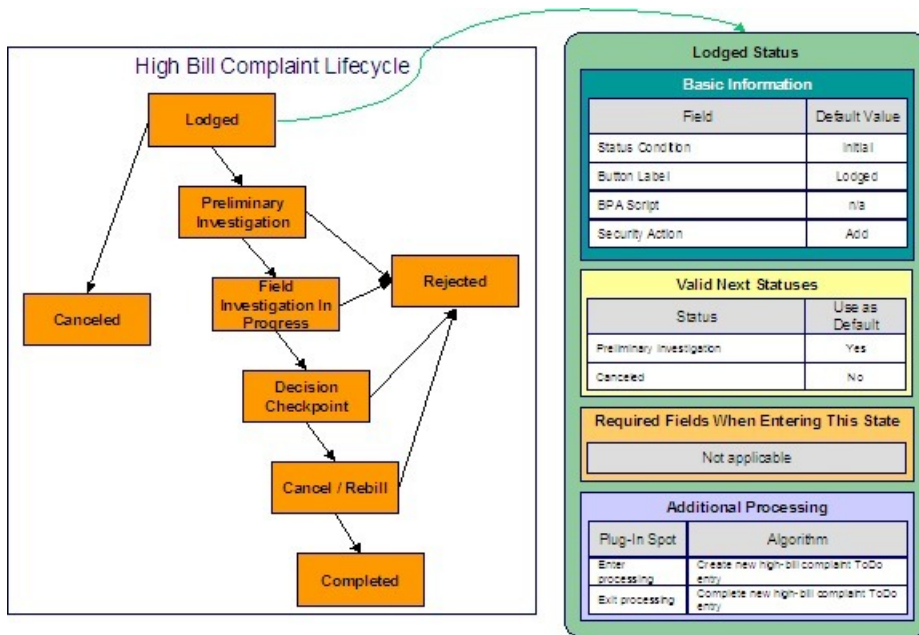
- The **Bill ID** characteristic is set up to be optional. This is because we've assumed that sometimes a high-bill complaint case can be lodged when you can't find the bill in question and you still want to log the case.
- Later in this section, you'll see that we've configured the Preliminary Investigation status to require a **Bill ID** before a case can enter this state.
- Both the **High Bill Complaint Cancel Reason** and **Reject Reason** are optional. Later in this section, you'll see that we've configured the Canceled and Rejected statuses to require these fields, respectively.
- Cases of this type do not need a **Responsible User** when first created. Later in this section, you'll see how we've configured the Preliminary Investigation status to require a **Responsible User** before a case can enter this state.
- Cases of this type do not need **Contact Information**. This was a subjective decision and depends on your organization's philosophy.

The topics that follow describe each of the statuses in a high-bill complaint's [lifecycle](#). We have assumed the following state transition rules:



Lodged High Bill Complaint

The following is an example of the configuration of the Lodged status for high bill complaint cases.

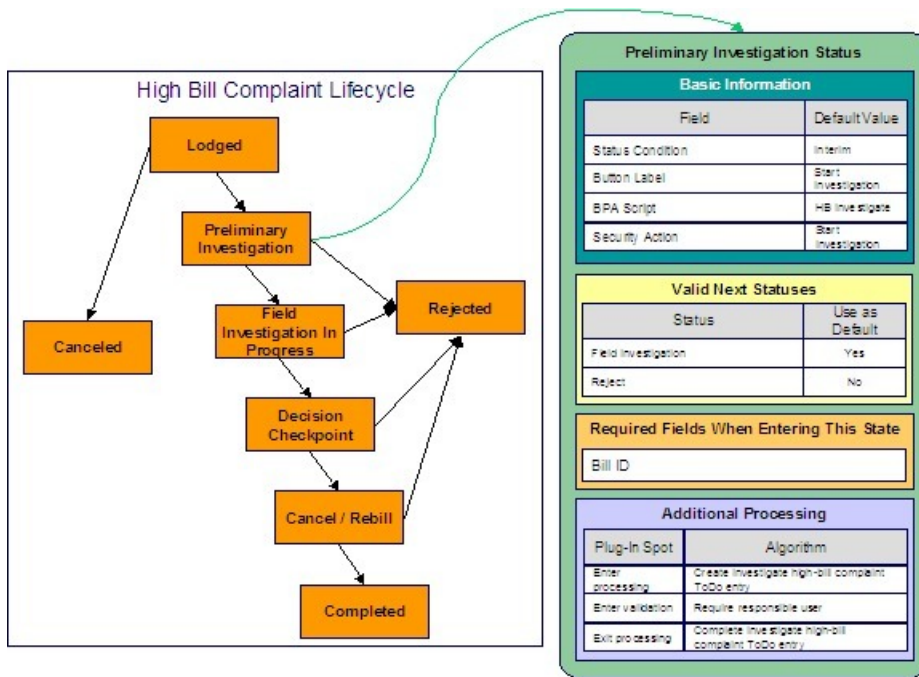


Note the following about this status:

- Lodged is the initial state. The initial state is the state populated on new cases of a given type. Remember that only one status can be defined as the initial state.
- It has a button label even though it's the initial state. The above diagram doesn't allow a user to ever transition a case into this state and therefore there will never be a button with such a label. However, it's a required field just in case you change your business rules.
- We have decided not to use a BPA script to help a user work on a high-bill complaint when it's in this state (this is probably not the best decision as BPA scripts can be quite useful).
- We have associated the Add action with this status. This means that only users with rights to the add action for the application service defined on the case type can add cases of this type.
- Notice that Valid Next Statuses are what restricts a case in this state to be transitioned to the Canceled and Preliminary Investigation states.
- Notice that the Additional Processing plug-ins create and complete a To Do entry when a case enters and exits this state, respectively.

Preliminary Investigation High Bill Complaint

The following is an example of the configuration of the Preliminary Investigation status for high bill complaint cases.

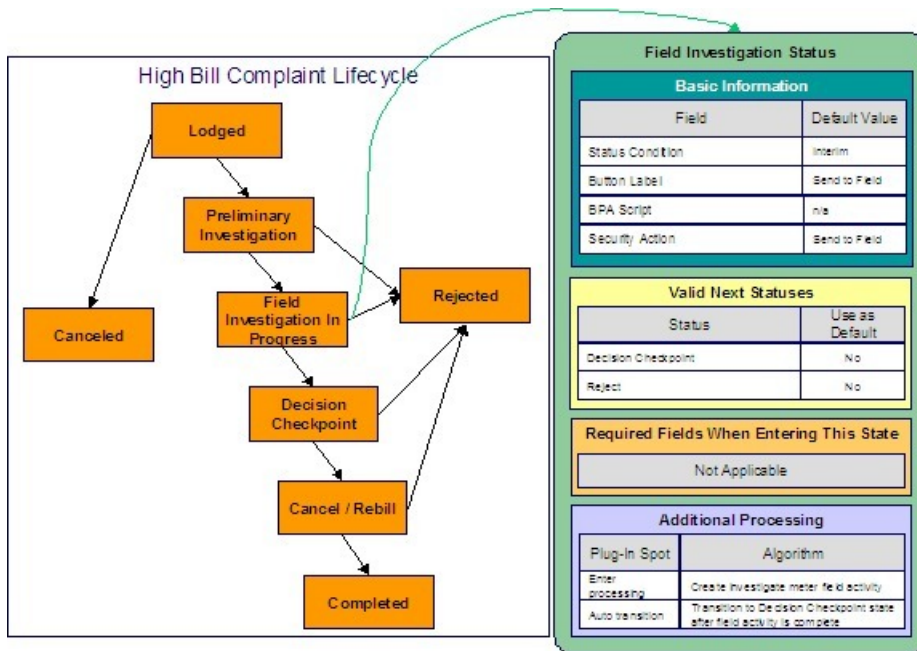


Note the following about this status:

- Preliminary Investigation is an interim state (meaning that it's not an initial or final state).
- It has a button label of **Start Investigation**. This is the label on the button that a user presses to move a case into this state. This button will only appear on cases that are in the Lodged state as this is the only state that can transition into the Preliminary Investigation state.
- We have decided not to specify a BPA script on this status. Rather, we're going to set up the To Do type used to highlight cases in this state to automatically launch an appropriate BPA script when a user selects the To Do entry.
- We have associated the Start Investigation action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state.
- Notice that Valid Next Statuses are what restricts a case in this state to be transitioned to the Field Investigation and Rejected states.
- Notice that a **Bill ID** must be specified on the case before it can be moved into this state.
- Notice that the Additional Processing plug-ins do the following:
 - Create and complete a To Do entry when a case enters and exits this state, respectively
 - Require a responsible user before a case can enter this state

Field Investigation High Bill Complaint

The following is an example of the configuration of the Field Investigation In Progress status for high bill complaint cases.

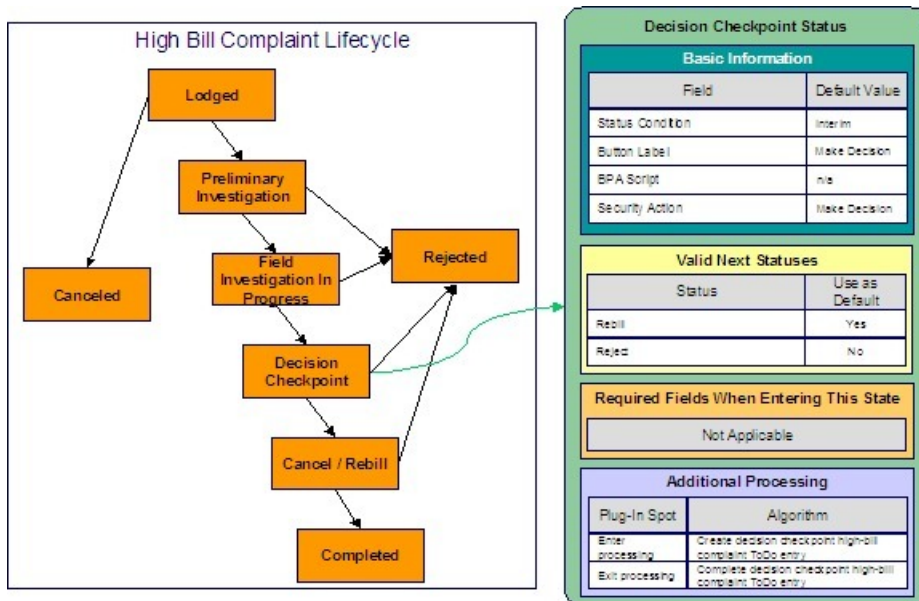


Note the following about this status:

- Field Investigation In Progress is an interim state (meaning that it's not an initial or final state).
- It has a button label of **Send to Field**. This is the label on the button that a user presses to move a case into this state. This button will only appear on cases that are in the Preliminary Investigation state as this is the only state that can transition into the Field Investigation in Progress state.
- We have decided not to specify a BPA script on this status because users don't work cases in this state (see the Additional Processing notes below for why this is the case).
- We have associated the Send to Field action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state.
- Notice that Valid Next Statuses are what restricts a case in this state to be transitioned to the Decision Checkpoint and Rejected states.
- Notice that the Additional Processing plug-ins do the following:
 - Create a field activity when a case enters this state
 - Cause the case to automatically transition to the Decision Checkpoint state when the field activity is completed

Decision Checkpoint High Bill Complaint

The following is an example of the configuration of the Decision Checkpoint status for high bill complaint cases.

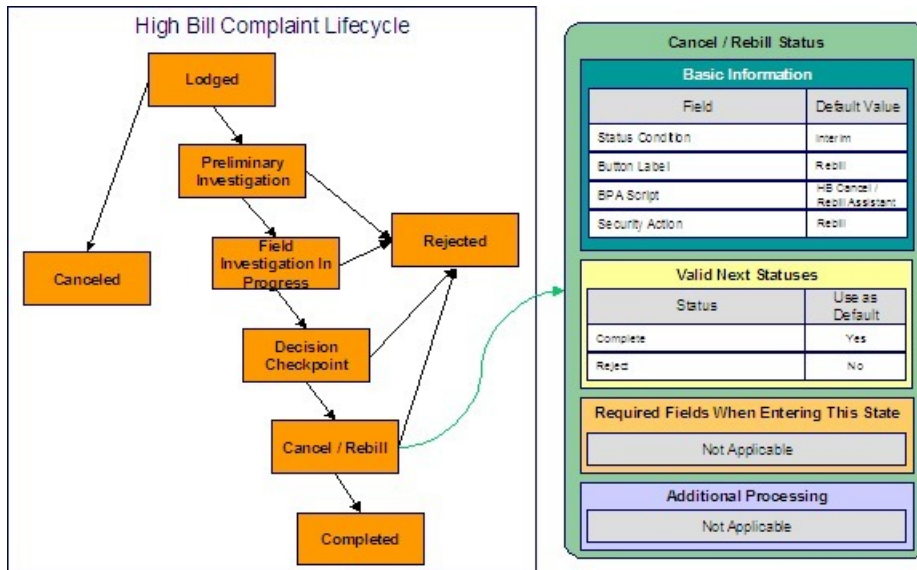


Note the following about this status:

- Decision Checkpoint is an interim state (meaning that it's not an initial or final state).
- It has a button label of **Make Decision**. This is the label on the button that a user presses to move a case into this state. This button will only appear on cases that are in the Field Investigation in Progress state as this is the only state that can transition into the Decision Checkpoint state.
- We have decided not to specify a BPA script on this status. Rather, we're going to set up the To Do type used to highlight cases in this state to automatically launch an appropriate BPA script when a user selects the To Do entry.
- We have associated the Make Decision action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state. Because the system will automatically transition cases into this state when the related field activity is complete, users will probably never press this button (and you may wish to prevent users from pressing this button by restricting security rights to the related action).
- Notice that Valid Next Statuses are what restricts a case in this state to be transitioned to the Cancel / Rebill and Rejected states.
- Notice that the Additional Processing plug-ins create and complete a To Do entry when a case enters and exits this state, respectively.

Cancel Rebill High Bill Complaint

The following is an example of the configuration of the Cancel / Rebill status for high bill complaint cases.

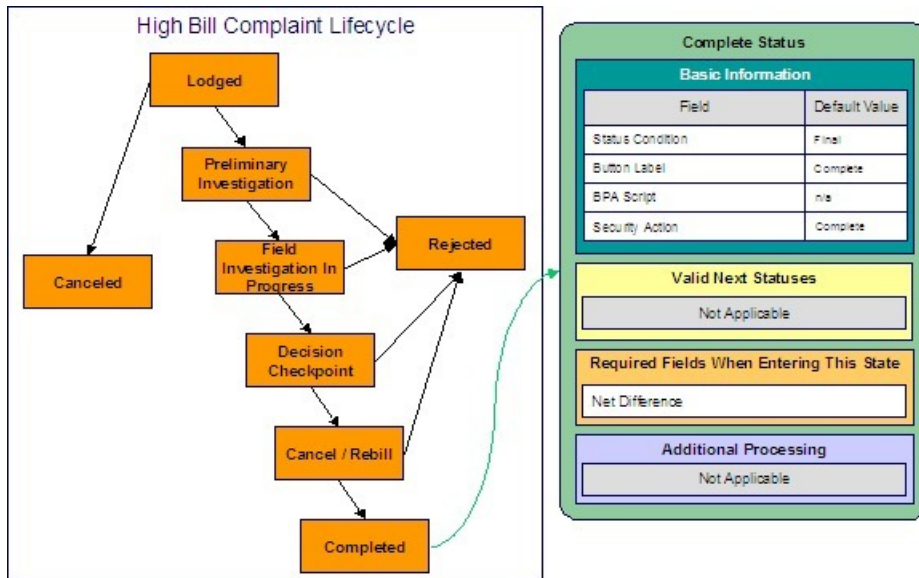


Note the following about this status:

- Cancel / Rebill is an interim state (meaning that it's not an initial or final state).
- It has a button label of **Cancel / Rebill**. This is the label on the button that a user presses to move a case into this state. This button will only appear on cases that are in the Decision Checkpoint state as this is the only state that can transition into the Cancel / Rebill state.
- We have referenced a BPA script that can assist a user in the cancel / rebill efforts.
- We have associated the Rebill action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state.
- Notice that Valid Next Statuses are what restricts a case in this state to be transitioned to the Completed and Rejected states.

Completed High Bill Complaint

The following is an example of the configuration of the Completed status for high bill complaint cases.

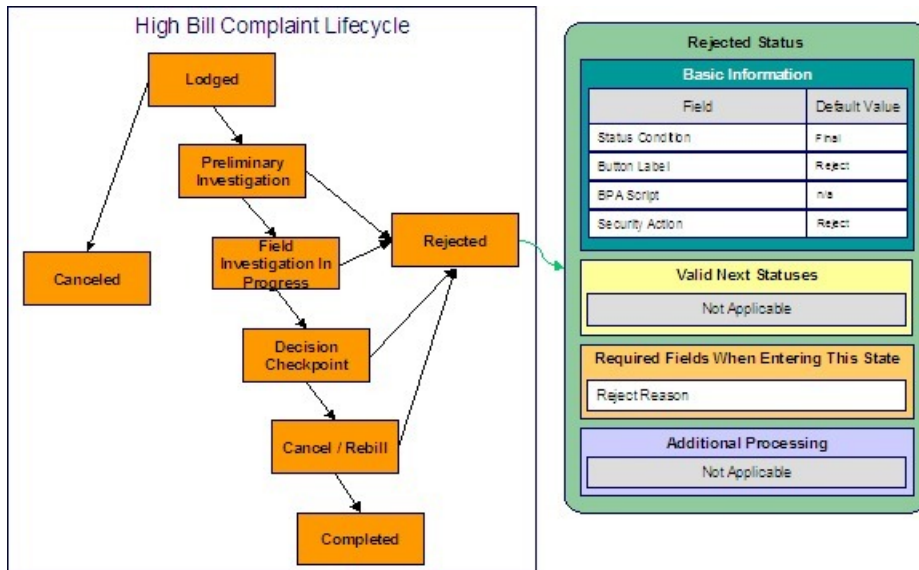


Note the following about this status:

- Completed is a final state (meaning that no further action is necessary).
- It has a button label of **Complete**. This is the label on the button that a user presses to move a case into this state. This button will only appear on cases that are in the Cancel / Rebill state as this is the only state that can transition into the Completed state.
- We have not referenced a BPA script because this is a final state (and no additional user action is necessary).
- We have associated the Complete action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state.
- Notice that there are no Valid Next Statuses (because this is a final state). If you wanted to allow a Completed case to be reopened, you would need to define the state into which a Completed case could be transitioned.
- Notice that the **Net Difference** must be specified on the case before it can be moved into this state. This would be the difference to the customer's balance after the cancel/rebill took place.

Rejected High Bill Complaint

The following is an example of the configuration of the Rejected status for high bill complaint cases.

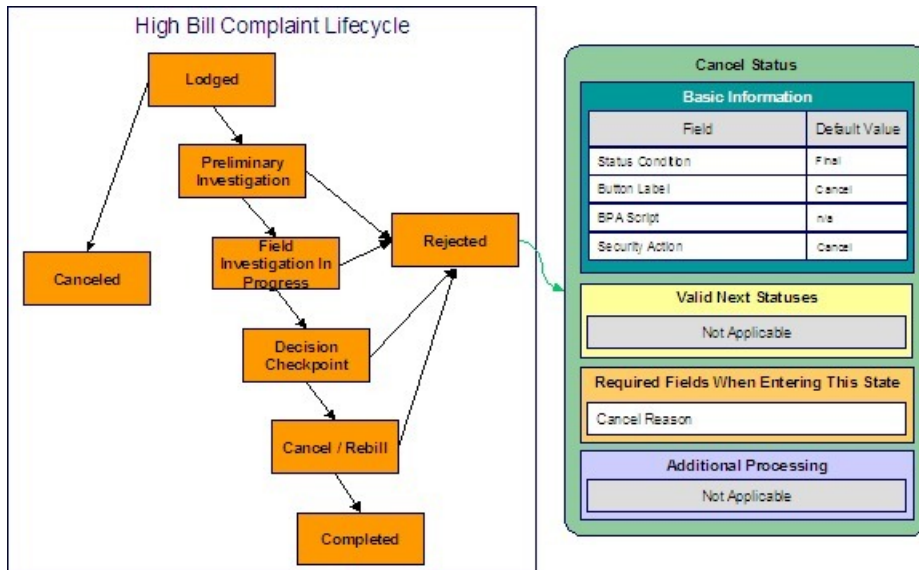


Note the following about this status:

- Rejected is a final state (meaning that no further action is necessary).
- It has a button label of **Reject**. This is the label on the button that a user presses to move a case into this state. This button will appear on cases that are in the Preliminary Investigation, Field Investigation in Progress , Decision Checkpoint and Cancel / Rebill states.
- We have not referenced a BPA script because this is a final state (and no additional user action is necessary).
- We have associated the Reject action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state.
- Notice that there are no Valid Next Statuses (because this is a final state). If you wanted to allow a Rejected case to be reopened, you would need to define the state into which a Rejected case could be transitioned.
- Notice that a **Reject Reason** must be specified on the case before it can be moved into this state.

Canceled High Bill Complaint

The following is an example of the configuration of the Canceled status for high bill complaint cases.



Note the following about this status:

- Canceled is a final state (meaning that no further action is necessary).
- It has a button label of **Cancel**. This is the label on the button that a user presses to move a case into this state. This button will only appear on cases that are in the Lodged state as this is the only state that can transition into the Completed state.
- We have not referenced a BPA script because this is a final state (and no additional user action is necessary).
- We have associated the Cancel action with this status. This means that only users with rights to this action for the application service defined on the case type can move a case into this state.
- Notice that there are no Valid Next Statuses (because this is a final state). If you wanted to allow a Canceled case to be reopened, you would need to define the state into which a Canceled case could be transitioned.
- Notice that a **Cancel Reason** must be specified on the case before it can be moved into this state.

Setting Up Case Management Options

The topics in this section describe how to set up the system to enable case management.

WARNING:

The following topics assume you thoroughly understand the concepts described under [The Big Picture Of Cases](#).

Installation Options

Case Info May Be Formatted By An Algorithm

An algorithm may be plugged in on the [installation record](#) to format the standard case information that appears throughout the system. Refer to [CSIN-DFLT](#) for an example of this algorithm.

This algorithm may be further overridden by a "Case information" plug-in on the [Case Type](#). Refer to [C1-CT-INFO](#) for an example of this algorithm.

Alert Info Is Controlled By An Installation Algorithm

An algorithm that is plugged in on the [installation record](#) is responsible for formatting the alerts that highlight if the person / account / premise in context has open cases. Refer to [CCAL-CASE](#) for an example of this algorithm.

Setting Up Application Services

As described under [Access Rights](#), you can prevent unauthorized users from accessing cases. The following points describe how to implement this type of security:

- Create an [application service](#) for each case type that needs to be secured
- Create an action on the application service for each status you need to secure
- Link the valid [user groups](#) to the application service and define which actions they can perform
- Define the application service on the [case type](#)
- Define the related action for each status on the [case type / status](#)

Setting Up Scripts

As described under [Scripts and Cases](#), BPA scripts can facilitate the creation and working of cases. Refer to the [Defining Script Options](#) for instructions describing how to set up scripts.

Setting Up To Do Types

As described under [To Do's and Cases](#), To Do entries can be used to highlight cases that require user attention.

The following points provide a high-level description of how to create (and complete) To Do entries for a case type:

- Create a To Do type for each different type of To Do entry used during a case's lifecycle
 - On the To Do type, think carefully about the roles whose users can work on the entries
 - Also consider if you would like a BPA script launched when a user selects the entry
- Specify the To Do type on the appropriate [entry processing](#) or [exit processing](#) algorithm
- If you want the system to automatically complete To Do entries, specify the To Do type on the appropriate [entry processing](#) or [exit processing](#) algorithm

Please be aware that the case maintenance transaction highlights the number of open and being worked To Do entries linked to the case being displayed on the page. However, the system can only do this if the To Do entries reference a [foreign-key characteristic](#) whose foreign key references the case table. If you use the [CSEN-TD](#) algorithm to create To Do entries when a case enters a given state, this algorithm will do this for you if:

- You have set up a [foreign-key characteristic type](#) whose [foreign key](#) references the case table
- In addition, the characteristic type must reference a characteristic entity of To Do Entry

Setting Up Characteristic Types

As described under [Additional Information](#), some of your cases may require additional information (in the form of [characteristics](#)). If this is true, you must set up the characteristic types before setting up the case types.

Refer to [Setting Up To Do Types](#) for instructions regarding a characteristic type that must be set up in order for the system to know the To Do entries that are associated with a case.

If you use the [CSEN-CC](#) algorithm to create customer contacts when a case enters a given state, you should set up a [foreign-key characteristic type](#) as follows:

- Its [foreign key](#) must reference the case table
- In addition, the characteristic type must reference a characteristic entity of Customer Contact

Setting Up Case Types

The case type maintenance transaction is used to maintain your case types. The topics in this section describe how to use this transaction.

FASTPATH:

Refer to [The Big Picture Of Cases](#) for more information about how a case type encapsulates the business rules that govern a case.

Case Type - Main

Use this page to define basic information about a case type.

Open the case type page by selecting **Admin > Case Type > Case Type > Add**.

NOTE:

Recommendation. Before using this transaction, we strongly recommend that you review the [Examples of Case Types](#).

Main Information

Enter a unique **Case Type** code and **Description** for the case type.

Use **Long Description** to provide a more detailed explanation of the purpose of the case type.

Person Usage controls the applicability of a person on cases of this type. Select Required if a person must be defined on this type of case. Select Optional if a person can optionally be defined on this type of case. Select Not Allowed if a person is not allowed on this type of case.

Account Usage controls the applicability of an account on cases of this type. Select Required if an account must be defined on this type of case. Select Optional if an account can optionally be defined on this type of case. Select Not Allowed if an account is not allowed on this type of case.

Premise Usage controls the applicability of a premise on cases of this type. Select Required if a premise must be defined on this type of case. Select Optional if a premise can optionally be defined on this type of case. Select Not Allowed if a premise is not allowed on this type of case.

If you need to restrict access to cases of this type to specific user groups, reference the appropriate **Application Service**. Refer to [Setting Up Application Services](#) for the details of how to secure access to your cases.

If you are configuring a case type to handle the processing of data defined via a **Business Object**, associating the case type with a business object serves to link the properties of the business object (e.g. BO options) with cases of that type. Refer to [Business Objects](#) for further information. In addition, refer to [Automatic Transition Rules](#) for information on the role of BO options in case auto-transition errors.

Responsible User Usage controls the applicability of a responsible user on cases of this type. Select Required if a responsible user must be defined on this type of case. Select Optional if a responsible user can optionally be defined on this type of case. Select Not Allowed if a responsible user is not allowed on this type of case. Refer to [Responsible User Applicability](#) for more information.

Contact Information Fields

There are three contact information fields: **Contact Person & Method Usage**, **Contact Instructions Usage**, and **Callback Phone Usage**. These fields are used to determine whether or not each type of contact information must be entered on case records with this case type. Select Required if the contact information must be entered, select Optional if the user can choose whether or not to include the contact information on this type of case, or select Not Allowed if the contact information cannot be entered on this type of case.

Algorithms

The **Algorithms** grid contains algorithms that control functions for cases of this type. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Case Information	Optional	<p>We use the term "Case information" to describe the basic information that appears throughout the system to describe a case. The data that appears in "case information" is constructed using this algorithm.</p> <p>Plug an algorithm into this spot to override the "Case information" algorithm on installation options or the system default "Case information" if no such algorithm is defined on installation options.</p> <p>Click here to see the algorithm types available for this system event.</p>

Case Type Tree

The tree summarizes the case type's lifecycle. You can use the hyperlinks to transfer you to the **Lifecycle** tab with the corresponding status displayed.

Case Type - Case Characteristics

To define characteristics that can be defined for cases of this type, open **Admin > Case Type > Search** and navigate to the **Case Characteristics** page.

Description of Page

Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on cases of this type. Turn on the **Default** switch to default the **Characteristic Type** when cases of this type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** box is checked. Refer to [Required Fields Before A Case Enters A State](#) for a description of how you can make option characteristics required at later stages in a case's lifecycle.

Case Type - Lifecycle

Case types that involve multiple users and multiple potential outcomes have complex lifecycles. Before you can design a case type's lifecycle, it's important that you thoroughly understand the concepts described under [Lifecycle](#) and [Status-Specific Business Rules](#). After thoroughly understanding these concepts, we recommend you perform the following design steps:

- Draw a "state transition diagram" as illustrated above under [Lifecycle](#). Keep in mind that if your state transition diagram is complex, your cases will be complex. While some cases warrant complexity, you should always ask yourself if there aren't better ways to achieve the desired results if your first effort results in complexity.
- Determine which characteristics (if any) are required during each stage of a case's lifecycle
- Determine when To Do entries (if any) should be created (and completed) during a case's lifecycle
- Determine additional validation (if any) that should be executed before a case enters and exits each state
- Determine additional processing (if any) that should transpire when a case enters or exits each state
- Determine if scripts are warranted to help users work the cases and, if so, design the scripts for each applicable state

When the above tasks are complete, you will be ready to set up a case type's lifecycle.

Open the Lifecycle page by selecting **Admin > Case Type > Case Type > Search** and navigate to the **Lifecycle** page.

NOTE:

You can navigate to a status by clicking on the respective node in the tree on the Main tab. You can also use the hyperlinks in the Next Statuses grid to display a specific status in the accordion.

Main Information

The **Status** accordion contains an entry for every status in the case type's [lifecycle](#).

Use **Status** to define the unique identifier of the status. This is NOT the status's description, it is simply the unique identifier used by the system.

Use **Description** to define the label that appears on the lifecycle accordion as well as the status displayed on the case.

Use **Script** to reference a BPA script that can assist a user work on a case while it's in this status. Refer to [A Script That Helps A User Work Through A Case](#) for the details.

Use **Access Mode** to define the action associated with this status. This field is disabled if an application service is not specified on the Main page. Refer to [Access Rights](#) for the details of how to use this field to restrict which users can transition a case into this state.

Use **Batch** to specify a batch control that will auto-transition the case. Any case in a status configured with a batch control will be transitioned when the batch job runs (rather than when [CASETRAN](#) is executed). For this purpose, batch process [C1-CSTRS \(Case Scheduled Transition\)](#) is supplied with base package, which will execute all Exit Status logic for the current status, and Enter Status logic for the destination status. You may choose to create a batch process with your own transition logic.

NOTE:

If you wish to defer transitioning a case in a particular status until the batch process on your case type status is executed, you should not populate an Auto-Transition algorithm on that status. Otherwise, CASETRAN will transition the case according to your Auto-Transition logic.

Use **Comment** to describe the status. This is for your internal documentation requirements.

Use **Sequence** to define the relative order of this status in the tree on the Main page.

Use **Status Condition** to define if this status is an Initial, Interim or Final state. Refer to [One Initial State and Multiple Final States](#) for more information about how this field is used.

Use **Transitory State** to indicate whether a case should ever exist in this state. Only Initial or Interim states can have a transitory state value of No.

The **Alert Flag** is used to indicate whether or not an alert should be displayed for customers with cases in the state. (The alert is shown via the base package Installation - Alert algorithm.)

Algorithms

The **Algorithms** grid contains algorithms that control important functions for cases of this type. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Description
Auto Transition	This algorithm is executed to determine if a case that's in this state should be transitioned into another state. Refer to Automatic Transition Rules for the details. Click here to see the algorithm types available for this system event.
Enter Processing	This algorithm holds additional processing that is executed when a case is transitioned into this state. You can also specify state transition logic within Enter Processing routines. Refer to Additional Processing When Entering A State for the details. Click here to see the algorithm types available for this system event.
Enter Validation	This algorithm holds validation logic that executes before a case can enter a given state. Refer to Validation Before A Case Enters A State for the details. Click here to see the algorithm types available for this system event.
Exit Processing	This algorithm holds additional processing that is executed when a case is transitioned out of this state. Refer to Additional Processing When Exiting A State for the details. Click here to see the algorithm types available for this system event.
Exit Validation	This algorithm holds validation logic that executes before a case can be transitioned out of a given state. Refer to Validation Before A Case Exits A State for the details. Click here to see the algorithm types available for this system event.

This algorithm sets the script launching option for the script associated with a given state, if any. Refer to [Script Launching Option](#) for the details.

Click [here](#) to see the algorithm types available for this system event.

Next Statuses

Use the **Next Statuses** grid to define the statuses a user can transition a case into while it's in this state. Refer to [Valid States versus State Transition Rules](#) for more information. Please note the following about this grid:

- Use **Action Label** to indicate the verbiage to display on the action button used to transition to this status.
- **Sequence** controls the order of the buttons that appear on [Case - Main](#). Refer to [Buttons Are Used To Transition A Case Into A State](#) for more information.
- **Use as Default** controls which button (if any) is the default button.
- **Transition Condition** may be configured to identify a common transition path for cases of this type in the current state. This transition condition may then be referenced across multiple case types. You'll need to add values to Look Up table field **TR_COND_FLG** that fit the typical transitions for your case types (e.g. Ok, Error, etc.).

By assigning the transition condition value to a given "next status", you can design your Enter State transition or Auto-Transition logic to utilize those flag values *without specifying a status particular to a given case type*. Thus, similar logic may be used across a range of case types to transition a case into, for example, the next Ok state for the case's current status.

- **Transition Role** controls whether only the System or both System and User have the ability to transition a case into a given "next status".
- You can use the status description hyperlink to open the Status accordion to the respective status.
- When you initially set up a case type, none of the statuses will reside on the database and therefore you can't use the search to define a "next status". We recommend working as follows to facilitate the definition of this information:
 - Leave the Next Statuses grid blank when you initially define a case type's statuses
 - After all statuses have been saved on the database, update each status to define its Next Statuses (this way, you can use the search to select the status).

Required Characteristics

Use the **Required Characteristics** grid to define characteristics that are required when a case enters this state. Only Optional characteristics defined on the main page appear in this grid. Refer to [Required Fields Before A Case Enters A State](#) for more information.

Workflow and Notification Options

We use the term "notification" to reference the electronic transactions that you exchange with third parties when:

- They need information about a customer
- They need to change something about a customer

For example, an energy service provider sends a notification to an electric distribution company when a customer elects to use them as their energy provider.

When a notification is received, the system responds by creating a workflow process. The workflow process contains workflow events. These events perform the processing necessary to execute the notification.

Workflow processing is difficult to explain because its flexible design can be used to automate many different types of multi-event processes. For example,

- You can use workflow processing to manage the events associated with the inspection of a new premise.

- You can use workflow processing to manage the events that transpire in a deregulated market when a customer wants to switch energy service providers.

WARNING:

Setting up the workflow process control tables is as challenging as your organization's business rules. If you don't have automated workflow processes, you don't have to setup anything. If you have sophisticated workflow processing requirements, your setup process will be more challenging.

The topics in this section describe tables that control your automated workflow and notification processing.

The Big Picture Of Workflow Processing

FASTPATH:

Refer to [The Lifecycle Of A Workflow Process And Its Events](#) for more information about workflow processes.

The Big Picture Of Workflow Events

This section describes the various types of workflow events and their lifecycle:

How Are Workflow Events Created?

Workflow events may be created as follows:

- The process that uploads notification requests creates a workflow process to implement the notification request. The workflow process has one or more workflow event(s). The number and type of events is controlled by the workflow process template associated with the workflow process. Refer to [What Type Of Workflow Process Is Created?](#) for more information about how notification requests cause workflow processes to be created.
 - Workflow events will be created when an operator creates an ad hoc workflow process. The number and type of workflow events are controlled by the workflow process template associated with the workflow process.
 - An ad hoc workflow event may be created and linked to an existing workflow process by an operator at their discretion.
 - The system may be customized to create a workflow process when something noteworthy happens in the system. Refer to [System Conditions May Trigger Notification and Workflow](#) for more information.
-

NOTE:

Bottom line. Most workflow events are created by the system when it creates a workflow process to implement an incoming notification. If you need to create an ad hoc workflow event, you can either create a workflow process using a template that contains the desired event OR link the desired event to an existing workflow process.

FASTPATH:

For more information about creating ad hoc workflow processes and events, refer to [Workflow Process Maintenance](#).

Executing Workflow Events On Their Trigger Date

When the background process (referred to as WFET) executes a workflow event on its trigger date, it calls the activation algorithm associated with the event's event type. Because you can add and change activation algorithms at will, the variety of workflow events is infinite.

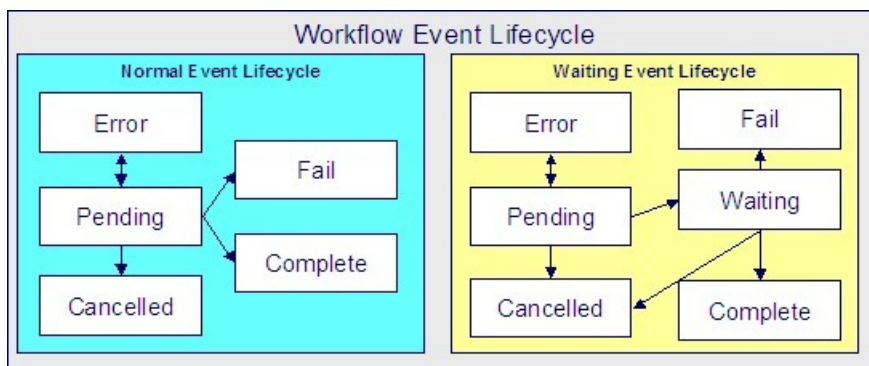
FASTPATH:

Refer to [Designing Workflow Event Types](#) for more information about activation algorithms.

Workflow Event Lifecycle

The lifecycle of a workflow event is dependent on whether the event has to wait on something before it can complete or fail. For example, an event that creates a field activity has to wait until the field activity is performed before it can be considered complete.

The following diagram shows the possible lifecycle of a workflow event:



The following points explain the lifecycle of workflow events of the *normal* variety:

- *Normal* events are initially created in the pending state.
- On a pending event's trigger date, the workflow event activation process (referred to as WFET) executes the event's activation algorithm.
- An event's activation algorithm may cause a pending event to become in error. For example, if an event used to create a field activity can't find a valid service point, the algorithm could change the status of the event to be in error. A user should correct the cause of the error and then change the event's status back to pending. The pending event will be processed the next time the activation program runs.
- An event's activation algorithm may cause a pending event to fail. For example, if an event used to validate a notification detects invalid information (e.g., an incoming notification is missing the customer's account number), the event will fail.
- A pending event becomes complete when the event's activity is successful. For example, if an event used to validate a notification determines the notification is valid, the event will complete. A user may manually change the status to complete if the event type indicates that manual completion is allowed.

FASTPATH:

For more information about a workflow event's trigger date, refer to [Workflow Event Dependencies & Trigger Date](#).

- A pending event will be cancelled automatically by the system if the workflow process is cancelled (a workflow process will be cancelled if an event fails or at the discretion of an operator). A user may cancel an event at will. Refer to [How Are Workflow Events Canceled](#) for more information.

The following points explain the lifecycle of workflow events of the *waiting* variety:

- *Waiting* events are initially created in the pending state.

- On a pending event's trigger date, the system executes the event. For example, if an event is used to create a "special read" field activity, on the event's trigger date, the field activity is created.
- An event's activation algorithm may cause a pending event to become in error. For example, if an event used to create a field activity can't find a valid service point, the algorithm could change the status of the event to be in error. A user should correct the cause of the error and then change the event's status back to pending. The pending event will be processed the next time the activation program runs.
- If the activation algorithm did not cause the event to become in error, the event's status is changed to waiting while the system waits for the field activity to be performed.

FASTPATH:

For more information about a workflow event's trigger date, refer to [Workflow Event Dependencies & Trigger Date](#). For more information about what an event might wait on, refer to [Waiting Events And Their Waiting Process](#).

- A waiting event becomes complete when the system sees that the thing that it's waiting for is finished.
- A waiting event fails when the system sees that the thing that it's waiting for didn't complete successfully. For example, an event that sends a confirmation request to a service provider asking if it's OK for a customer to switch suppliers would fail if the service provider denies the request.
- A waiting event will be cancelled automatically by the system if the workflow process is cancelled (a workflow process will be cancelled if an event fails or at the discretion of an operator).
- A pending event will be cancelled automatically by the system if the workflow process is cancelled (a workflow process will be cancelled if an event fails or at the discretion of an operator).

FASTPATH:

Refer to [Workflow Processes Can Have Multiple Branches](#) for more information about event transition in a process with multiple branches.

Workflow Event Dependencies & Trigger Date

The trigger date of most workflow events is blank when they are first created. This is because the trigger date can only be set when ALL of the preceding workflow events on which it depends are complete. An example will help explain why this design is necessary. Consider the following example that shows a simple workflow process and its events:

Event Number	Workflow Event	Dependent On Event(s)	Trigger Date Set To X Calendar Days After Completion Of Preceding Events
10	Validate new customer notification	N/A - first event	0
20	Set up new customer and meter	10	0
30	Send notification confirming new customer	20	0
40	Send welcome letter	20	0

This workflow process is meant to implement the following:

- On the first day, validate the incoming notification (the one that tells us about a new customer).
- If validation is successful, set up the new customer and their meter in the system.

- After the new customer and meter are set up, send a notification to the requester that everything has been set up. Also, send a letter to the customer.

The problem is that you don't want to execute event 20 until event 10 is complete. This is achieved by indicating event 20 is dependent on event 10. The system will only populate event 20's trigger date when event 10 is complete. Similarly, you can't set the trigger dates of events 30 and 40 until the customer has been set up (event 20). So, when you set up a workflow event, you must indicate the dependent events. If you only want the next event to trigger X days after the completion of earlier events, you can indicate such.

NOTE:

Bottom line. The trigger date of a workflow event is set to the current date plus X days where X is the number of calendar days defined on the workflow event. If this date is not a workday for your organization, the trigger date will be set to the next workday. If the resultant date is the current date (because X is zero), the event will be activated immediately.

FASTPATH:

Refer to [How Are Workflow Events Completed?](#) for information about how these events are executed.

Workflow Events May Be In Error

As explained under [Workflow Event Lifecycle](#), when the background process WFET executes an event's activation algorithm, this algorithm may cause a pending event to become in error. For example, if an event used to create a field activity can't find a valid service point, the algorithm could change the status of the event to be in error.

For every workflow event that's in error, a record is written to the [workflow event exception](#) table. To view the errors,

A user should correct the cause of the error and then change the event's status back to pending. The pending event will be processed the next time the activation process runs.

FASTPATH:

Refer to [Errors versus Failure](#) for information to help you differentiate between events that have failed versus those that are in error.

Some Workflow Events May Fail

Some workflow events may fail. For example:

- An event that validates an incoming notification may result in failure if the notification contains invalid information.
- An event that asks for the confirmation from a distribution company may fail if the distribution company rejects the request.

FASTPATH:

Refer to [Errors versus Failure](#) for information to help you differentiate between events that have failed versus those that are in error.

The background process that is responsible for activating events (referred to as WFET) is the process that can cause an event to fail (failure can happen during the activation algorithm on the workflow event). When an event fails, the system:

- Updates the workflow process with message number and message parameters describing the validation problem.
- Sets the status of the event to Failed.

- Cancels the workflow process and its outstanding events.
- Calls the failure algorithm defined on the event's event type.

It's important to note that some types of events can't fail and therefore don't have a failure algorithm. For example, an event that creates a field activity can't fail, neither can an event that creates a welcome letter.

FASTPATH:

Refer to [Workflow Processes Can Have Multiple Branches](#) for more information about event failure in a process with multiple branches.

Errors versus Failure

As explained under [Workflow Event Lifecycle](#), an event's activation algorithm may cause a pending event to become in error or to fail (amongst other things). You control the exact state when you design your workflow event type activation algorithms.

The main differences between these two states is as follows:

- As described under [Some Workflow Events May Fail](#), a failed event causes the entire workflow process to fail (and it cannot be restarted).
- As described under [Workflow Events May Be In Error](#), a user can correct the cause of an error event's error and then change the event's status back to pending. The pending event will be processed the next time the activation process runs.

You should follow the following guidelines when designing your validation logic in your workflow event activation algorithms:

- If the cause of the problem is correctable by a user, you should set the state of the event to be in error.
- If the cause of the problem is not correctable by a user (e.g., you were interfaced information that cannot be corrected by your users), you should set the state of the event to fail.

Waiting Events And Their Waiting Process

Some events have to wait until something else happens before they can be Completed (or Fail). These types of events exist in the Wait state until the thing they are waiting for happens. For example, consider an event that creates a field activity - it has to wait until the field activity is complete before it can itself Complete.

Every type of event that waits for something else to happen before it completes or fails must have a corresponding background process that monitors the thing on which the event is waiting. We refer to background processes that perform this monitoring as Waiting Processes.

There will be a Waiting Process for every type of event that has to wait for something to happen. The specific background process is defined on the workflow event type. For more information, refer to [Designing Workflow Event Types](#).

The following points describe the responsibilities of a Waiting Process:

- Check on the thing on which the event is waiting. For example,
 - An event that creates a field activity has a Waiting Process that checks on the state of the field activity.
 - An event that creates a request to confirm a customer's request to switch suppliers has a Waiting Process that checks if the confirmation is accepted or rejected.
- Change the state of the event to Complete or Fail based on what transpired. For example,
 - When the field activity completes, the Waiting event can Complete
 - The acceptance or rejection is received, the Waiting event can Complete or Fail

- Detect that the event has been waiting too long and do something. For example, the Waiting Process could:
 - Create a To Do entry (this might be useful if you need an operator to do something)
 - or, create an outgoing notification informing the sender of the incoming notification that something is wrong
 - or, Fail / Complete the event (you may be able to automatically assume success or failure if an event waits longer than a predefined limit)
 - or, execute the event again and reset the base time on the event (this might be useful if the event initiates an outgoing notification to ask permission from some other service provider - if it waits too long, you could simply create another outgoing notification)
 - or, whatever else you can think of developing in the process

How Are Workflow Events Canceled?

The background process that is responsible for activating events (referred to as WFET) automatically cancels workflow events when an event fails.

A pending event will be cancelled automatically by the system if the workflow process is cancelled (a workflow process will be cancelled if an event fails or at the discretion of an operator).

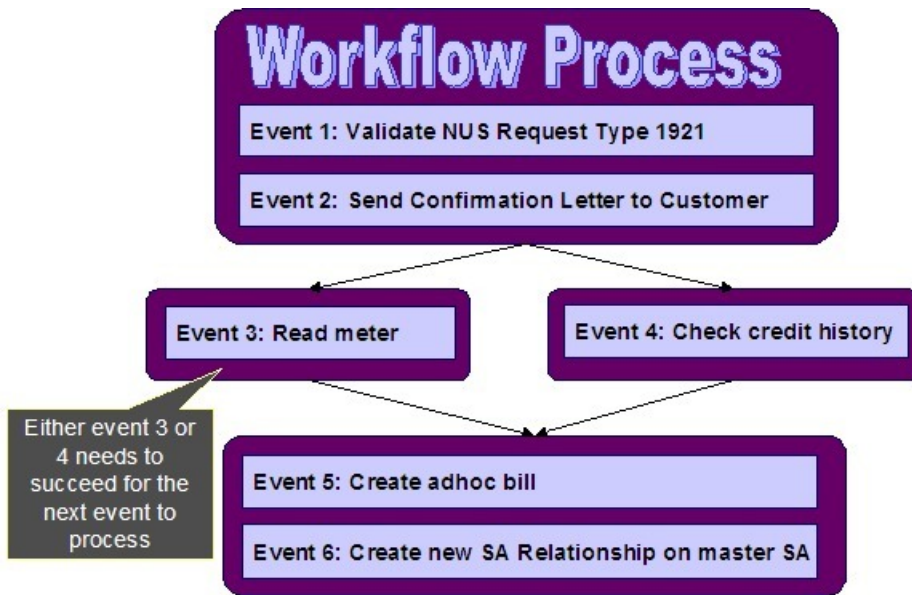
In addition, an operator may cancel a workflow event at will.

An event will be cancelled by the background process that is responsible for activating events (referred to as WFET) when it detects that ALL of its earlier, dependent events are cancelled. This is important to understand if your organization has [Workflow Processes With Multiple Branches](#).

Workflow Processes Can Have Multiple Branches

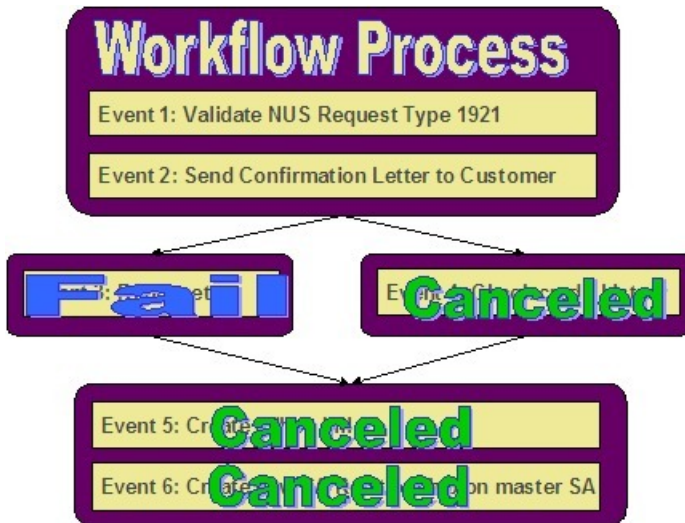
Using the workflow event dependencies, a workflow process may have multiple branches. There are several reasons why you may want to set up a process to contain multiple branches:

- There may be events that can run in parallel. This is useful if the related tasks take time to execute. For example "Read Meter" and "Send Confirmation and Wait for Response". Both can be executed in parallel and event 5 will execute based on the outcome of events 3 and 4. The following is an example of such a workflow process.

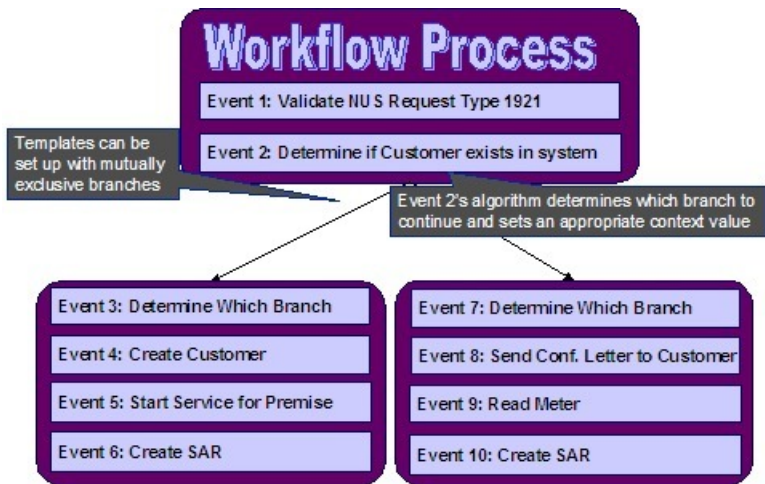


In the example above, if event 3 is canceled and event 4 completes, (or if event 4 is canceled and event 3 completes), event 5 will proceed. However, if both event 3 and event 4 are canceled, all remaining events will be canceled.

It should be noted that if either event 3 or 4 fails, ALL events in the process will be canceled, including events in a different branch.



- You may have a business process that has some common events and some events that are mutually exclusive. Rather than setting up several processes, you can set up one process that branches based on specific criteria. For example, perhaps you get to a certain point in a process that differs based on whether or not a meter is installed. If there is a meter, you follow one branch and if there is no meter, you follow a different branch.

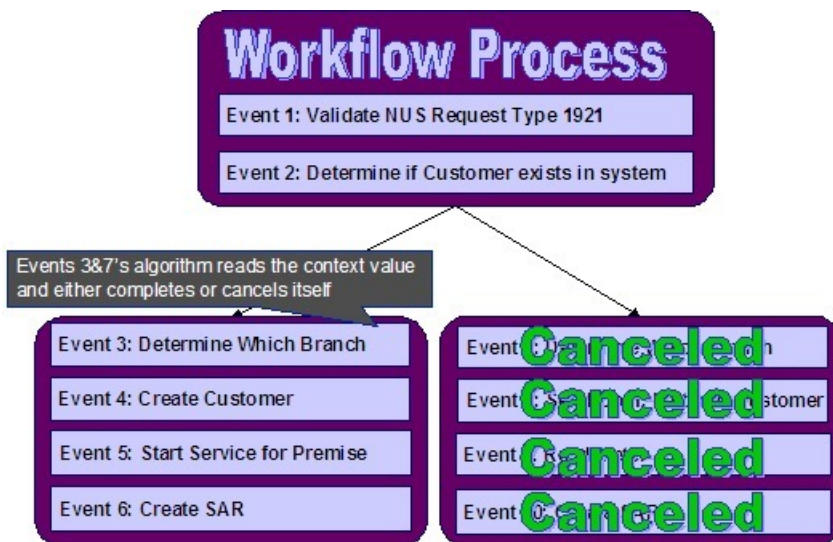


In the above example, event 2 determines if the workflow process is associated with an existing customer or a new customer.

- For a new customer; events 3, 4, 5 and 6 are executed.
- For an existing customer, events 7, 8, 9 and 10 are executed.

Event 2 does this by creating an entry in the Context collection indicating which branch should continue. (For example Context Type / Value of BRANCH / A). Events 3 and 7 should be special events whose purpose is to read the BRANCH context type and determine if its branch should continue or be canceled.

In this example, Event 3's algorithm will continue if the context value for BRANCH is A and Event 7 will only continue if the context value for BRANCH is B.



NOTE:

The above diagram and description indicates the use of context records to determine which branch to continue. If your implementation chooses to use characteristics instead, the same logic above may be implemented using characteristics as well.

The Big Picture of Notification Processing

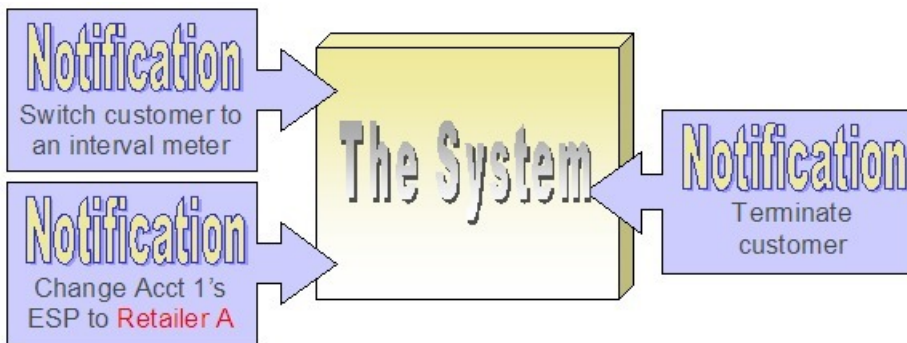
We use the term "notification" to reference the electronic transactions that you exchange with third parties when:

- They need information about a customer.
- They need to change something about a customer.

For example, an electric service provider may send a notification to an electric distribution company when a customer elects to use them as their energy provider.

You may have to support a wide variety of notifications. For example:

- You may receive a notification that tells you to switch a customer's energy service provider to a different provider (if you are a distribution company).
- You may receive a notification that asks for the last 24 months of consumption history for a customer (if you maintain consumption).
- Etc.



When a notification is received, the system responds by creating a workflow process. The related workflow process contains workflow events and it is the events that actually do whatever needs to be done (e.g., change the customer's service provider, create field activities, stop service, etc.).

In addition to incoming notifications, the system creates outgoing notifications when:

- It needs to respond to an incoming notification. For example, if an incoming notification requests a customer's consumption history, the system must send the consumption history by creating an outgoing notification.
- It needs to apprise a third party that something has changed about the customer. For example, if a customer stops service, the system must tell the various service providers of such.

The topics in this section describe how notifications are interfaced into and out of the system.

Uploading Notifications Into The System

The following diagram illustrates the steps involved in the uploading of notifications into the system.



When a notification's electronic transaction is received, it must be inserted into the notification upload staging (NUS) table. Why? Because the system periodically looks for pending records in the NUS table and attempts to create a workflow process to implement the notification's request.

It's important to be aware that the events in a workflow process execute the notification request. To help solidify this concept, consider this - the first event in a workflow process typically validates the attributes on the NUS record.

FASTPATH:

Refer to [Notification Upload Background Process](#) for more information about the upload process.

What Type Of Workflow Process Is Created?

Given that you can receive many different types of notifications (e.g., return consumption, switch supplier, stop service), it should make sense that each type of notification upload staging (NUS) record will probably require a different type of workflow process to implement its request. The system uses the NUS record sender's external system Id and a notification upload type to determine the type of workflow process. This works as follows:

- Every external system references a workflow process profile.
- A workflow process profile defines the type of workflow process template to use to process a given notification upload type.

NOTE:

Different workflow processes for the same type of notification. It's probably obvious why different notification types result in different workflow processes. You may wonder why different workflow processes would be needed by two senders who submit the same type of notification? The reason is that different senders may have different types of computer systems that handle their notification processing (or no computer system at all). For example, you may have to respond by fax to a sender who doesn't have a sophisticated computer system, whereas you may send an electronic transaction to a sender who is technically advanced. The system allows you to create different workflow process profiles for each response mechanism (e.g., sophisticated vs. fax). You then associate the appropriate workflow process with each sender.

How Are Notifications Sent Out Of The System?

In addition to processing incoming notifications from third parties, the system also sends notifications to third parties to provide them with information (and to ask them for information). It is also possible to use notifications to send information to other applications within your company.

A Notification Download Staging (NDS) record is created for every notification that is sent to the outside world. NDS records are created by workflow events (i.e., the event's activation algorithm creates a NDS record). Consider the following examples:

- Many workflow processes exist to process incoming notifications. These processes typically contain workflow events that create NDS records to communicate with service providers. If you look at the workflow processes described under [Designing Workflow Process Templates](#), you will see many such examples.
- When something noteworthy happens, the system may need to tell a third party about it. Or perhaps you have third party software, which contains information from the system and your company needs to keep the information in sync.

FASTPATH:

Refer to [System Conditions May Trigger Notification and Workflow](#) for more information about triggering notification download staging records from within the system.

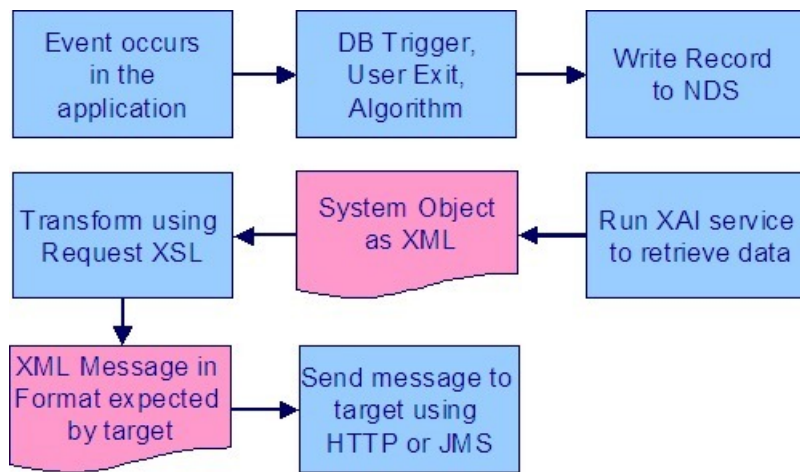
Notification and XAI

NDS records may also be processed using the XAI tool. Refer to [Outgoing Messages](#) for an overview of communicating outgoing messages via XAI.

The remaining sections describe the capability to send outgoing messages from the system to another application using the Notification and Workflow engine and XAI.

Near-Real Time Outgoing Messages

The following diagram illustrates the flow of near-real time messages:



The following points describe the diagram

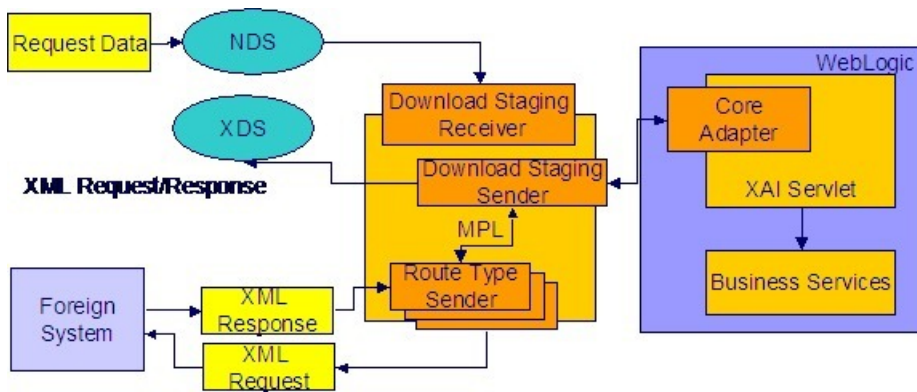
- When an event occurs in the system, you can initiate an outgoing message by storing a [notification download staging](#) (NDS) record. NDS records for outgoing messages have a special processing method flag of XAI.

FASTPATH:

Refer to [System Conditions May Trigger Notification and Workflow](#) for information about creating notification download staging records.

- Once a message is stored on the NDS table, the download staging receiver reads it and invokes XAI to extract data from your product. An XAI inbound service is used to retrieve the full information for the object related to the message.
- The download staging sender takes the response from the executor (the system object as XML) and continues the processing. It uses a request XSL on the route type to transform the message to a format expected by the target. It then sends the message to the target using the sender on the XAI route type.

The following diagram illustrates the process:



The responsibilities of the download staging receiver and download staging sender are provided in detail below.

- [Download Receiver](#)
- [Download Staging Sender](#)
- [Asynchronous Responses to NDS Messages](#)

Download Receiver

The download staging [receiver](#) processes records in the NDS table that have a processing method flag equal to XAI and a status of either pending or retry . It also uses the priority flag on the NDS type and the creation date/time on the NDS to process records according to their priority.

The process is referred to as "near real time" because your download staging receiver should be continuously checking for new records in the NDS table.

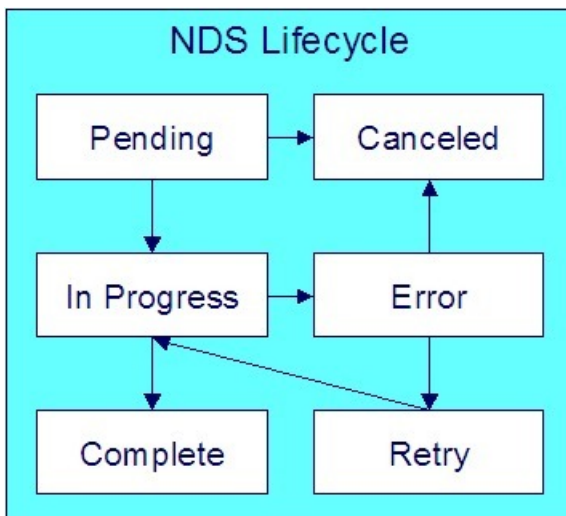
When it finds a record, it invokes XAI (via the [executer](#)) providing the XAI inbound service.

The following section provides more information about the NDS lifecycle.

- [Lifecycle of Notification Download Staging](#)
- [Building the XML Request](#)

Lifecycle of Notification Download Staging

The download receiver processes NDS records based on their status. The following diagram describes the lifecycle of an XAI type NDS.



The following points describe the diagram:

- Records are created in pending status. A user may transition the record to the canceled state.
- The MPL processes records in pending and retry status. While the record is being processed, it is changed to in progress.
- Based on the results of the processing, records could be transitioned to error or complete.

NOTE:

An NDS is set to error if there is a problem with the NDS or if any of its related XDS records are in error.

- A user may cancel a record in the error state or change the status to retry to have the MPL try again.

Building the XML Request

To build the XML request for the outgoing message, the download receiver needs the following information:

- The XML request and response schemas used to retrieve information about the record related to the request. For example, if the outgoing message is to inform an external system that a person's name has changed, the XML request schema must retrieve the appropriate person record.
The [NDS type](#) related to the NDS defines an XAI Inbound Service that references an appropriate XML request schema.
- The data related to the request. In this example, the NDS record should be stored with the ID of the person record that changed.
The person ID is stored as NDS context, with an appropriate Context Type, like PERID.
- XPATH information to tell the system where to plug in the related data into the request schema. In this example, when building the XML request, the person id must be plugged into the appropriate location in the XML request schema prior to executing the request.
The [NDS type](#) related to this NDS defines a collection of context types. For each context type, you define an [XPATH](#) used to indicate where to substitute the context data.

Refer to [Configuring the System for NDS Messages](#) for more information.

Once the request schema is build, the receiver invokes XAI (via the executer) to execute the XML request. In our example, it calls the appropriate Person service and the XML response includes the extracted person information.

[Top of the Page](#)

Download Staging Sender

This piece is somewhat confusing, because the download staging sender behaves differently than other senders. [Senders](#) are typically responsible for

- Routing a request to an appropriate target
- In the case of the upload staging sender and the staging control sender, the sender is responsible for updating the status of the appropriate table.
In the case of the download staging sender, it's responsible for processing the "response" to the XAI executer, whose task was to build the XML request.
- If there is a problem executing the request, the download staging sender updates the NDS record in error and creates an [NDS exception](#) record.
- If the executer was successful, the "response" is an XML request in a format understood by your product. The sender must perform the following steps to help process the outgoing message:
- Transform the request to a format expected by the target system(s)
- Route the outgoing message to the target
- Create an XAI download staging record for each message sent out. (There is typically only one, but it is possible for the outgoing message to have [multiple destinations](#).)

If any errors are found during any of these steps, the status of the NDS record is set to error and a record is written to the [NDS exception](#) table. If the error is related to routing the message, the XDS is created in error and a record is written to the [XDS exception](#) table.

NOTE:

Configuration required. The above explanation assumes that you have correctly configured your download staging receiver to reference the download staging sender. Refer to [Designing Responses for a Receiver](#) for more information.

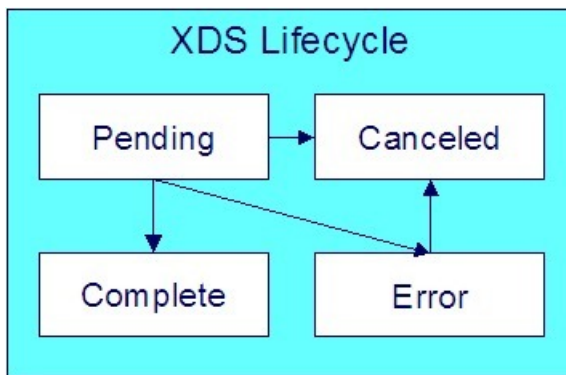
Routing Optional. It is possible that an [NDS message does not require routing](#) information. In this case the download staging sender simply updates the NDS record's status

The following sections provide further detail.

- [Lifecycle of XAI Download Staging](#)
- [Transform the Request](#)
- [Routing the Message](#)
- [Multiple Message Destinations](#)
- [An NDS Message That Doesn't Require Routing](#)

Lifecycle of XAI Download Staging

The following diagram illustrates the lifecycle of an XAI download staging record.



An XAI download staging record is created in one of the following ways:

- When the download staging sender continues the processing for the outgoing message, it attempts to transform and route a message for each XAI route type indicated on the notification download profile. An XDS is created to record the message sent.
- If the message was sent successfully, the XDS is created in complete status.
- If an error occurred when sending the message, the XDS is created in error status and an [XAI download exception](#) is created.

NOTE:

Automatic Resend. If you have configured the system for [automatic resend](#) and the system detects that the error is due to the sender being unavailable, no XDS record is created in this case.

- When the system attempts to send a message [real time](#) but the external system is unavailable, the system may create a pending NDS and a pending XDS record.
- A user may cancel a pending or error XDS record.

If you have resolved the error for an XAI download staging record, change the status of the related NDS record to retry. When MPL processes the NDS again, it deletes all XDS records in error and attempts to send them again.

Transform the Request

The download staging sender takes the XML response with the extracted data and attempts to transform into a format accepted by the target. It needs an appropriate [XSL transformation](#) script.

The XSLT is defined on an [XAI Route Type](#), which is referenced on the [download profile](#) of the service provider referenced by the notification download staging record.

NOTE:

The NDS context information is available to the XSL.

Routing the Message

In addition to defining the appropriate XSL transformation scripts, the XAI route type also references a [sender](#) that tells the system how to route the message, for example it may be routed using a JMS queue.

NOTE:

Inactive Senders. If the message is targeted to an inactive sender and you have enabled [automatic resend](#), the message is ignored and an MPL log entry is created indicating that the message was not sent.

If the target supports synchronous communication (for example, an HTTP sender), the [route type](#) may be configured to receive an acknowledgement. This indicates to the download staging sender to wait for a response from the sender on the route type.

The download staging sender creates an appropriate XDS record with the XML request document and with a status of complete or error based on the results of the communication with the target. If the target has sent a synchronous response, the XML response is also posted on the XDS.

NOTE:

Automatic Resend. If you have configured the system for [automatic resend](#) and the system detects that the error is due to the sender being unavailable, no XDS record is created in this case and the NDS remains in pending.

If the target supports synchronous communication, you may indicate that a response to the outgoing message also requires action. For example, perhaps the outgoing message to an external system causes a new record to be created in that system. The response to the message may require an update to our initiating record to post the external system's identifier for the record on their side.

To enable this logic, you must check the Post Response switch on the [route type](#). If this has been checked, the download staging sender creates an XAI upload staging record (and a staging control record) and it links the XDS record as a foreign key. The XUS record is processed along with other uploaded messages, to invoke an appropriate service.

NOTE:

XML Response. If the Post Response switch on the route type is turned on, the XML response that is captured is the XML that results after applying the Response XSL. If the Post Response switch is not turned on (i.e., the response is a simple acknowledgement), XML response is posted without applying a Response XSL. (In fact, for simple acknowledgements, there is no need for a Response XSL.)

Asynchronous Responses. Refer to [Asynchronous Responses to NDS Messages](#) for information about processing responses to messages sent asynchronously.

FASTPATH:

Refer to [Configuring the System for NDS Messages](#) for more information about setting up these controls.

Multiple Message Destinations

If your message must be sent to more than one external system, a different XAI route type is required for each system that the message must be sent to so that the sender and XSLT for each external system can be defined.

The recommended procedure for sending a message to multiple destinations is to

- Generate an NDS for each destination where each NDS defines the same NDS type but different service providers. In other words, the service provider represents the destination.
- Each service provider references a [notification download profile](#) where the appropriate XAI route type is defined for each NDS type.
- When this NDS record is processed, the sender information defined for the XAI route type is used to route the message appropriately and an XAI download staging (XDS) record is created.

Note that it is possible for the download profile's formatting method to reference more than one XAI route type. This is perhaps another configuration that could be used to send a message to multiple destinations. In this case an XDS record is created for each XAI route type to track the XML request. However, it is not possible to track [asynchronous responses](#) to multiple messages using this mechanism. Defining multiple route types for a download profile formatting method should only be done if you do not expect an asynchronous response.

An NDS Message That Doesn't Require Routing

If an NDS message is being sent to one of your own external systems and this system may be accessed directly, you could design your message such that XAI routing is not necessary. For this scenario, the assumption is that the application service that is invoked by the XAI inbound service completes all the necessary steps to update the external system.

If this is the case, you are not required to define an XAI Route Type on the appropriate notification download profile entry. In addition, no message sender is required for this external system.

The download staging sender's sole responsibility for a message of this type is to update the NDS status.

[Top of the Page](#)

Asynchronous Responses to NDS Messages

If you send an asynchronous NDS message and you expect a response, the response is received as an inbound message. In order to recognize that an inbound message is a response to an NDS message, there must be some handshaking. In other words our system must pass a unique identifier of our message to the external system and the external system must include that identifier in the response.

- [Populating the Unique Message Identifier](#)
- [Recognizing the Inbound Message Response](#)

Populating the Unique Message Identifier

As described in [Building the XML Request](#), the information that may be included in the XML request we build is information linked to the NDS record as context. Therefore, the unique identifier you want to send to the external system must be stored as a context record. (Let's call this context type Message ID). Because the notification download staging ID is a unique identifier, you may choose to populate Message ID with the NDS ID. However, it is possible that the external system requires an identifier in a different format. For example, the NDS ID is a 12-byte number. Perhaps the external system can only receive a 10-byte number for the Message ID.

It is the responsibility of the mechanism that creates the NDS record (i.e., the algorithm or user exit or database trigger) to create a context entry for Message ID with an appropriate value that is compatible with the target system.

When creating your XML request schema, if you expect a response to your message, you must include the message ID as an attribute in the XML request. Because the message id is not part of the product information that your request schema is extracting, it should be defined as a private attribute. Once the XML is created, it is the responsibility of the XSL to transform the Message ID into the target specific XML element.

NOTE:

Only One Route Type. Because the unique Message ID for an NDS message is linked to the NDS record via context, it is not possible for the system to handle multiple route types (and therefore multiple senders) for a single message IF an

asynchronous response is expected. The reason is because all senders receive the same unique Message ID and if each sender responds, the system is not able to determine which response is received for which sender.

Recognizing the Inbound Message Response

When the external system sends an asynchronous response to your NDS message, it is received as an [inbound message](#). This inbound message must include the Message ID that you sent so that you can recognize this as a response. In addition, the XSL transformation must populate the identifier of the external system sending the message.

The following points describe the detail related to receiving an inbound message:

- When a response comes in, the message is translated using an XSLT script. The XSLT must map the message ID, which is the unique identifier of the NDS message that this is a response to, to the XML element MessageID. It must create an attribute with the XML element ReplyToMessageSource populating the value with an appropriate [external system](#). This value should correspond to the external system code of the service provider related to the NDS message.
- The XML request of the inbound message is processed as normal.
- The existence of the special message elements in the XSL indicates to XAI that XML response processing is required. It looks for a complete NDS with a service provider whose external id matches the ReplyToMessageSource and with a context type of Message ID whose context value matches XAI Message ID.
- When the correct NDS is found, the system finds the related XAI download staging record and populates the XML response with the XML request of the incoming message.
- If the XDS request is not associated already with a response or the latest associated response is in error status stamp the current response on the XDS request record.
- If the response came into the system as an XAI upload staging (XUS) record, the foreign key to the XDS record (NDS ID and XAI route type) to which it is responding is linked to the XUS.
- If the response did not come in as an XAI upload staging record (for example, it came in via JMS queue), an XAI upload staging record and an XAI staging control record are created in complete status as an audit and the foreign key to the XDS record is linked to the XUS.

NOTE:

Errors Linking the XDS. If the system has a problem finding the appropriate XDS record to link to the incoming response, the inbound message is still processed. A log entry is added to the XAI trace file reporting the error in identifying the correct XDS record.

Real Time Outgoing Messages

This section describes the capability to send real time messages from the system to another application. The functionality works in conjunction with the near real time NDS message configuration.

Real Time Message Engine

The system provides an "engine" to communicate real time messages to an external system. This engine may be invoked by a user exit on any user interface in the system. The real time message engine receives an NDS type and a service provider as input. Using this information, the engine finds the appropriate XAI route type for the service provider's notification download profile. The route type indicates the XSLT and the XAI sender.

NOTE:

HTTP Sender. In the current release of the product, only senders that communicate via HTTP are supported.

Unlike near real time messages, the XAI inbound service referenced on the NDS type is not used to build the XML request. Rather, the user exit that invokes the engine is responsible for extracting the appropriate data and passing to the engine

a formatted XML document. The XSLT referenced on the route type should map attributes from XML document into a format expected by the target system.

The following points describe how the real time engine works:

- Once the XML request is built, the message is sent out based on the XAI route type's sender information.
- The system waits for a response for a predefined time out limit defined as a context entry on the [sender](#).
- When the response is received, the engine transforms the message using the response XSL defined on the route type.
- The XML response is then passed back to the user exit that called the engine. The user exit processes the response as needed.

NOTE:

As described in *Routing the Message*, the route type includes a flag to indicate to the system if a response requires additional action. If so, an XUS is created. The real time message engine does not support this logic. It expects the user exit to process the response as needed.

Sending Real Time Messages in Near Real Time

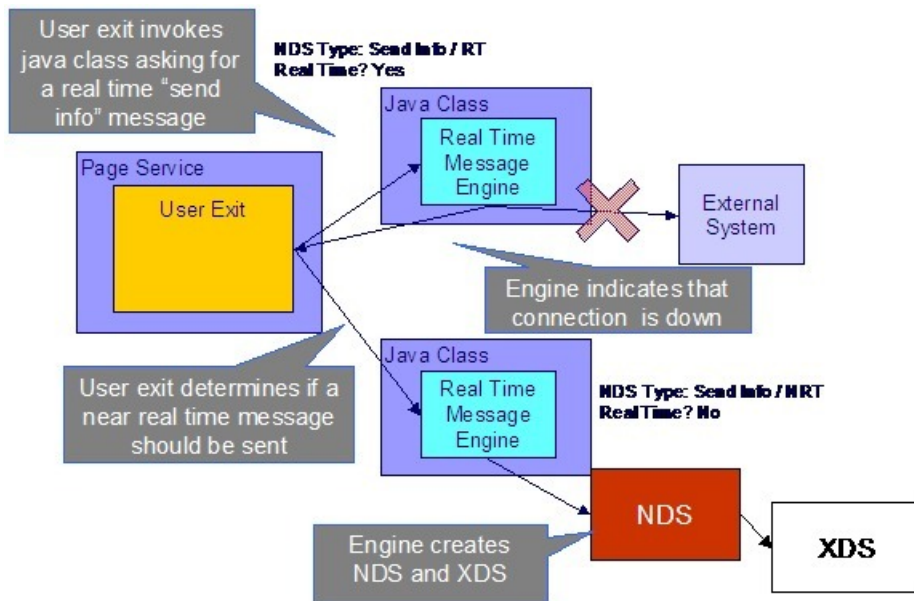
If the connection is not available when the real time engine attempts to communicate with the target, the engine may post the message to the download staging tables to send in "near real time" when the target system is available again.

Messages should only be posted to the staging table if the user does not need to wait for a response in real time. Consider the following examples:

- A real time message is used to request information from an external system to display to a user on a page. If the connection is down, it does not make sense to store a near real time message for this case because the user is not able to wait for the response to the request.
- There is a connection problem when the user attempts to send information to an external system. In this case, perhaps it makes sense to store a near real time message so that the information is sent to the external system when the connection is restored.

NOTE:

The real time message engine does not make the decision to post a message near real time when the communication is unavailable. The responsibility lives solely with the user exit that calls the java class that invokes the engine. The following diagram illustrates a possible interaction between the user exit and the message engine:



The following points describe the diagram:

- When the user exit is informed that the real time message could not be sent, it is the responsibility of the user exit to decide if the engine should be invoked again to create a near real time message.
- The NDS type used for near real time messages is different than the one used for real time messages. The reason is because the XSL transformation script is different for real time vs. near real time.
- The NDS is created and marked pending.
- The XDS record is created and marked pending. The XML message that is ready to be sent to the sender is posted as the XML request for the XDS.

Because the XML request is already built, the NDS type used for the NDS should reference a special XAI inbound service. This service basically tells the download staging sender that the XDS already exists and doesn't need to be created.

System Conditions May Trigger Notification and Workflow

NOTE:

Outbound Messages. The mechanism for triggering communication to an external system described here does not take advantage of the configurable business object functionality that enables outbound messages to be triggered from a server-based script. Refer to [Outbound Messages](#) for more information about communicating with an external system using outbound messages.

There may be many noteworthy occurrences in the system, which should trigger the notification and workflow process.

Some of these occurrences may need to be communicated to an external system. Examples of when this may occur are as follows:

- You use an external software package, which uses information from the system and you need to send a message to this software package when certain attributes of the customer change.
- You need to inform third parties when something changes about a customer. For example, if you are a distribution company, you probably advise a customer's energy supply company when their meter is changed or when the customer stops. Or if a customer, who has been referred to a collection agency, makes a payment, you will need to inform the agency of the payment.

In many scenarios, you may want more than just a notification to be sent out, but you would also require a workflow process. For example, perhaps when a customer's mailing address changes, you want to a) send a confirmation letter to the customer's old address and their new address; b) inform a third party service provider about the change; and c) send the update to an external marketing system. In this case, the act of changing the mailing address could trigger a workflow process, which would perform the above steps.

You may have other scenarios where a noteworthy condition in the system should trigger a workflow process, which does not require a notification to be sent out of the system. For example, perhaps your company follows a sophisticated procedure for starting service, which requires certain events to transpire before other events can occur. You could create a workflow process, which gets triggered upon starting service to handle this sophistication.

In summary, when something occurs in the system, you may need to

- Create a notification download staging (NDS) record. You would do this if you only need to send a message to an external system.
- Create a notification upload staging (NUS) record. You would do this if you need to trigger a workflow process to do one or more steps. One of the workflow events in this process could create a NDS if an external system needs to be informed.

Creating a Notification Download Staging Record

To create an NDS record directly, you need the following information:

- Service Provider (who will receive the outbound message)
- NDS Type

You also need to populate relevant data using either context or characteristics. For example, if a change to a person's name should trigger an outbound message, the Person ID must be referenced.

The process that creates the NDS must determine the appropriate service provider that should receive this information based on the business rules. As for the NDS type, this is typically "hard-coded" based on what has occurred to trigger the outbound message. For example, if the customer's name changed, then the NDS Type is set to something like "Name Change".

For any NDS records automatically created by the system in sample algorithms or in system processes, the NDS type is not hard-coded. Rather, the system uses a download type condition flag value to look up the appropriate NDS type. As a result, if an implementation chooses to use these sample algorithms, but prefers to use a different NDS type, it is possible. When designing your own NDS types that are referenced by NDS records created based on system conditions, you may choose to create new download type condition flag values as well.

Refer to [Designing Notification Download Types](#) and [Setting Up Notification Download Types](#) for more information.

Creating a Notification Upload Staging Record

To create an NUS record directly, you need the following information:

- External System (to point to a service provider)
- NUS Type

You also need to populate relevant data using either context or characteristics. For example, if a change to a person's name should trigger a letter to the customer and an outbound message, the Person ID must be referenced.

The process that creates the NUS must determine the appropriate external system based on the business rules. If the steps that need to be taken as a result of this system condition are based on an internal practice, the external system could be something like "SYSTEM". That external system would reference an appropriate workflow process profile that defines the appropriate workflow process based on the NUS type.

Determining the NUS type is similar to determining the NDS type as described in the previous section.

Refer to [Designing Notification Upload Types](#) and [Setting Up Notification Upload Types](#) for more information.

Triggering Notification & Workflow from the Application

You and your implementers must determine the noteworthy events that should trigger workflow and/or outgoing notifications based on your business practice. For example, you may communicate all meter exchanges and customer address changes to the customer's various service providers.

Next, you must assess how you are going to create the appropriate records:

- Determine if there is a plug-in spot that exists where you can create an algorithm to trigger the NUS or NDS. For example, if canceling a service agreement should trigger a notification, an SA Cancel algorithm plugged in to the SA type may be used.
- If there is no plug-in spot, determine if there is a user exit at the appropriate place for you to trigger the creation of an NUS or NDS
- You may also decide to use a database trigger to create the NUS or NDS

Once you have determined how and when to trigger the creation of notification records, have your implementers create the CM code to do the work.

The Lifecycle of Notification Download Staging

If a download staging record is routed to the outside world via XAI, a status is assigned to the download staging record.

Refer to [Lifecycle of Notification Download Staging](#) for more information.

Creating Notification and Workflow Procedures

A workflow process is created by copying the events defined on a workflow process template (a workflow process template contains the standard events).

NOTE:

Workflow processes can be used for any multi-event process. While this section describes how to design workflow processes to support incoming and outgoing notifications, workflow processes can be used for other multi-step processes. For example, you can use a workflow process to manage the steps involved with the installation and inspection of backflow devices at a water service point.

The topics in this section describe how to design and setup the tables that control your notification and workflow processing.

FASTPATH:

For more information about notification and workflow processing, see [The Big Picture Of Workflow Events](#) and [The Big Picture Of Notification Processing](#).

WARNING:

There are innumerable ways to design your notification and workflow procedures. Some designs will result in easy long-term maintenance, others will result in maintenance headaches. In this section, we provide information to help you

understand the ramifications of the various options. Before you set up your production procedures, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

Designing Notification Upload & Workflow Procedures

The topics in this section describe how to design the tables that control your notification upload and workflow processing.

Designing Workflow Process Profiles

The following sections describe how to design a workflow process profile. A workflow process profile controls the type of workflow process that will be created to process your incoming notifications. If you plan to use workflow processes outside of the notification upload process, you can skip to [Designing Workflow Process Templates](#).

You associate a workflow process profile with one or more external system. Whenever a notification is received from an external system, the system uses the workflow process profile to determine the type of workflow process to create to implement the notification.

The easiest way to design a workflow process profile is to choose a representative external system and design a workflow process profile for it by filling in the matrices below. After you've designed a profile, determine how many other external systems can use it. Then design the next external system's profile and determine who can reuse it. Repeat this process until all your external systems have a profile. Once the profiles are designed, you're ready to set up the workflow control tables.

FASTPATH:

Refer to [The Big Picture Of Notification Processing](#) for more information about how workflow processes are used to implement incoming notifications.

The topics discussed below will gradually complete the following matrix using a simple case-study. We recommend that you use the following matrix as your guide. When the matrix is complete, you're ready to set up a workflow process profile.

WF Process Criteria
Notif. Upload Type

Designing Notification Upload Types

You will need one notification upload type for every type of notification your organization can receive from the various service providers. To "design" your notification upload types, you document the codes used by external systems to identify the transaction types on their notification upload staging records.

We have populated the Notification Upload Type column with a few classic examples that can be received by a distribution company:

WF Process Criteria
Notif. Upload Type
Retrieve a customer's consumption history
Switch a customer to a different service supplier

NOTE:

It is recommended that the unique identifier of your Notification Upload Types match the "transaction types" that are referenced on incoming notifications. If it is not possible to do this, the process that creates the upload staging records must map the external transaction types to the notification upload types that you define in the system.

In addition, as described in [System Conditions May Trigger Notification and Workflow](#), you should evaluate any condition that should cause an NUS to be created (to eventually create a workflow). A new NUS Type should be defined for each condition. For each of these, consider creating a new value for the upload condition flag so that the NUS type is not hard-coded in the system process that creates the NUS.

Navigating to Related NUS Extension

When designing the types of notifications your organization may receive, you must determine where you plan to store the detailed information related to the incoming transaction: as context linked to the NUS, as characteristics linked to the NUS or by using an extension record.

FASTPATH:

Refer to [Process X - Populate Notification Upload Staging](#) for information about the different options available.

If you choose to capture the detailed information in a new extension record with a new user interface, then in order to be able to drill down to the related extension record, the following steps are required.

- Add a new [lookup](#) value for your new extension record for the field NT_UP_EXTSN_FLG. (This step is needed regardless of whether you want to drill down to the related record.)
- Add a [navigation option](#) pointing to the appropriate navigation key for your new transaction. This navigation option must reference a usage type of notification upload type.
- Indicate the appropriate lookup value and navigation option on the appropriate notification upload type.

Designing Workflow Process Criteria

The matrix's second dimension is dependent on workflow criteria algorithms. Workflow criteria are confusing. Think of them as optional conditions that, if met, will cause a different type of workflow process to be started when a given notification upload type is received.

You must define a Default criterion in case none of the override criteria are met. You MAY have override criteria if different situations result in different types of workflow processes. For example, let's assume some notification upload types have different workflow processes for industrial customers as compared to all other types of customer. This assumption necessitates the introduction of an override workflow process criteria; we'll call it Industrial Customer.

WF Process Criteria	Default	Industrial Customer
Notif. Upload Type		
Retrieve a customer's consumption history		
Switch a customer to a different service supplier		
Switch customer to an interval meter		

NOTE:

The workflow process criteria are limited by your imagination (and business requirements). We have provided the workflow process criteria you see above as an example; we don't expect you'll be able to use the exact conditions we supply. Your conditions will be based on any number of factors.

New workflow process criteria may require programming. See [How To Add A New Algorithm](#) for more information.

Designing Workflow Processes To Process Incoming Notifications

The next step involves populating each cell in the matrix with the workflow events that should be executed when the system receives a notification identified with a given notification upload type. If override criteria aren't relevant for a given notification upload type, we will mark the cell as "N/A".

WF Process Criteria	Default	Industrial Customer
Notif. Upload Type		
Retrieve a customer's consumption history	Validate notification - consumption history request Confirm requester is a valid service provider for the customer's service. Create notification download - send consumption history	N/A (meaning that industrial customers use the Default criteria)
Switch a customer to a different service supplier	Validate notification - supplier switch Confirm requester is a valid service provider for the customer's service. Check with current supplier if the switch is allowed. Switch suppliers.	N/A (meaning that industrial customers use the Default criteria)
Switch customer to an interval meter	Validate notification - change customer to an interval meter Create notification download - reject request (only industrial customers can have an interval meter)	Validate notification - change customer to an interval meter Confirm requester is a valid service provider for the customer's service. Create field activity to exchange current meter. Change rate on exchange date.

NOTE:

Notice that the first event in each cell typically validates the notification upload staging record.

At this point, the matrix is complete. Before you're ready to design your workflow process templates you must design the processes described in the next section.

Designing Workflow Processes To Deal With Invalid Sender or Notification Upload Type

The system needs to know the type of workflow process to create when a notification is uploaded that does not contain a valid External System or Notification Upload Type (these two fields are the ones that control the type of workflow process that's created to process the uploaded notification). When these conditions are detected by the notification upload process, most utilities create an outgoing notification rejecting the uploaded notification when such conditions transpire. We've shown these in the following table.

Notification Condition	Workflow Process Events
Unknown Notification ID	Create notification download - reject notification, bad external system
Unknown Notification (Upload) Type	Create notification download - reject notification, bad notification upload type

NOTE:

These processes are not in a workflow process profile. The above workflow process templates are not referenced in a workflow process profile. Rather, when you create these workflow process templates you label them with a **Notification Condition** of Unknown Notification ID or Unknown Notification Type. The notification upload process will then create workflow process when these events transpire.

Designing Workflow Process Templates

The following table shows the workflow process templates referenced in the previous section's matrix. Adjacent to each process is its events and an indication of when they are triggered.

Workflow Process Template	Event Number	Workflow Event Type	Dependent On Event(s)	Trigger Date Set To X Calendar Days After Completion Of Dependent Events
Retrieve a customer's consumption	10	Validate notification - consumption history request	N/A - first event	0
	20	Confirm requester is a valid service provider for the customer's service	10	0
	30	Create notification download - send consumption history	20	0
Switch a customer to a different service supplier	10	Validate notification - supplier switch	N/A - first event	0
	20	Confirm requester is a valid service provider for the customer's service	10	0
	30	Check with current supplier if the switch is allowed	20	0
	40	Switch suppliers	30	0
Switch customer to an interval meter	10	Validate notification - interval meter switch	N/A - first event	0

	20	Confirm requester is a valid service provider for the customer's service	10	0
	30	Create field activity to exchange current meter	20	0
	40	Change rate on exchange date	30	0
Reject interval meter switch	10	Validate notification - interval meter switch	N/A - first event	0
	20	Create notification download - reject request	10	0
Reject bad notification upload type	10	Create notification download - reject request	N/A - first event	0
Reject bad external system	10	Create notification download - reject request	N/A - first event	0

NOTE:

The workflow process for "Reject bad notification upload type" should reference the Notification Condition Unknown Notification Type. The workflow process for "Reject bad external system" should reference the Notification Condition Unknown Notification ID.

Designing Workflow Event Types

If we extract each unique event type from the above table, we end up with the following:

Workflow Event Type

Validate notification - consumption history request

Confirm requester is a valid service provider for the customer's service

Create notification download - send consumption history

Validate notification - supplier switch

Current supplier confirmation

Switch supplier

Validate notification - interval meter switch

Create field activity to exchange meter

Change rate

Create notification download - reject request

Next, we have to determine the algorithm that will be used when each event is activated on its trigger date. We call this algorithm the *activation algorithm*. An activation algorithm is a stand-alone routine that does whatever you need done when an event is activated. We have populated the following table with brief descriptions of the types of activation algorithms you'd need for the above workflow events.

NOTE:

The activation algorithms are limited by your imagination (and business requirements). We have provided the activation algorithms you see below as an example; we don't expect you'll be able to use the exact algorithms that we

supply. Your algorithms will be based on any number of factors. Be aware that new activation algorithms may require programming. See [How To Add A New Algorithm](#) for more information.

Workflow Event Type	Activation Algorithm
Validate notification - consumption history request	Validate consumption history request
Confirm requester is a valid service provider for the customer's service	Confirm service provider is valid requester
Create notification download - send consumption history	Create notification download - send consumption history
Validate notification - supplier switch	Validate supplier switch request
Current supplier confirmation	Create notification download - check if it's OK to switch customer from current supplier
Switch supplier	Switch supplier
Validate notification - interval meter switch	Validate interval meter switch request
Create field activity to exchange meter	Create field activity - exchange meter
Change rate	Change rate
Create notification download - reject request	Create notification download - reject request

Next, we have to determine which types of events can fail. Refer to [Some Workflow Events May Fail](#) for background information failure. For those types of events that can fail, we will indicate their *failure algorithm* in the table. A failure algorithm is a stand-alone routine that does whatever you need none when an event fails. We have populated the following table with brief descriptions of the types of failure algorithms you'd need for the above workflow events.

NOTE:

The failure algorithms are limited by your imagination (and business requirements). We have provided the failure algorithms you see below as an example; we don't expect you'll be able to use the exact algorithms that we supply. Your algorithms will be based on any number of factors. Be aware that new failure algorithms may require programming. See [How To Add A New Algorithm](#) for more information.

Workflow Event Type	Activation Algorithm	Failure Algorithm
Validate notification - consumption history request	Validate consumption history request	Create notification download - invalid request
Confirm requester is a valid service provider for the customer's service	Confirm service provider is valid requester	Create notification download - invalid requester
Create notification download - send consumption history	Create notification download - send consumption history	N/A
Validate notification - supplier switch	Validate supplier switch request	Create notification download - invalid request
Current supplier confirmation	Create notification download - check if it's OK to switch customer from current supplier	Create notification download - reject request due to supplier rejection
Switch supplier	Switch supplier	N/A
Validate notification - interval meter switch	Validate interval meter switch request	Create notification download - invalid request
Create field activity to exchange meter	Create field activity - exchange meter	N/A
Change rate	Change rate	N/A
Create notification download - reject request	Create notification download - reject request	N/A

And finally, for those events whose activation algorithm puts them into a wait state, we have to determine the waiting process that monitors the waiting events. Refer to [Waiting Events And Their Waiting Process](#) for more information about waiting.

NOTE:

The waiting processes are limited by your imagination (and business requirements). We have provided the waiting processes you see below as an example; you may not be able to use the exact processes that we supply as your processes will be based on any number of factors. Be aware that a new waiting process will require programming.

Workflow Event Type	Activation Algorithm	Failure Algorithm	Waiting Process
Validate notification - consumption history request	Validate consumption history request	Create notification download - invalid request	N/A
Confirm requester is a valid service provider for the customer's service	Confirm service provider is valid requester	Create notification download - invalid requester	N/A
Create notification download - send consumption history	Create notification download - send consumption history	N/A	N/A
Validate notification - supplier switch	Validate supplier switch request	Create notification download - invalid request	N/A
Current supplier confirmation	Create notification download - check if it's OK to switch customer from current supplier	Create notification download - reject request due to supplier rejection	Check if supplier has accepted process
Switch supplier	Switch supplier	N/A	N/A
Validate notification - interval meter switch	Validate interval meter switch request	Create notification download - invalid request	N/A
Create field activity to exchange meter	Create field activity - exchange meter	N/A	Check if field activity is complete
Change rate	Change rate	N/A	N/A
Create notification download - reject request	Create notification download - reject request	N/A	N/A

Designing External Systems

As described under [What Type Of Workflow Process Is Created?](#), every notification upload staging (NUS) record contains an external system. An external system is the unique identifier of the sender of the notification.

To "design" your external systems, document the external id's (e.g., DUNS number) of your service providers. The following table contains a sample:

External System	Description
102910	Energy Supplier, Inc.
392191	Cheap Power, Inc.

The above example would result in 2 external systems - 102910 and 392191.

NOTE:

It's obvious, but worth stressing, that the External System should be the exact number referenced on notifications sent by a sender.

Designing Notification Downloads

The system creates outgoing notifications when:

- It needs to respond to an incoming notification. For example, if an incoming notification requests a customer's consumption history, the system must send the consumption history by creating an outgoing notification.
- It needs to apprise a third party that something has changed about the customer or their meter. For example, if a customer stops service, the system must tell the various service providers of such.
- It needs to exchange information with another application in the company, for example a CRM system.

A Notification Download Staging (NDS) record is created for every notification that is sent to the outside world.

The topics in this section describe how to design the tables that control your notification download processing.

Designing Workflow Processes To Support Outgoing Notifications

As described under [How Are Notifications Sent Out Of The System?](#) notifications are sent out via Notification Download Staging (NDS) records. NDS records are created by workflow events (i.e., the event's activation algorithm creates notification download records). There are two types of workflow processes that contain these types of workflow events:

- Many workflow processes exist to process incoming notifications. These processes typically contain workflow events that create –NDS records to communicate with service providers. If you look at the workflow processes described under [Designing Workflow Process Templates](#), you will see many such examples.
- When something noteworthy happens, the system may need to tell a third party about it. As described in [System Conditions May Trigger Notification and Workflow](#), you may decide that the noteworthy condition should cause a workflow process to be created where one or more of the workflow events can create an NDS. The following table shows representative workflow process templates of this type.

Workflow Process Template	Event Number	Workflow Event Type	Dependent On Event(s)	Trigger Date Set To X Days After Completion Of Dependent Events
Stop service	10	Create notification download - inform all parties that customer is stopping service	N/A - first event	0
Meter exchange	10	Create notification download - inform all parties that meter has been changed	N/A - first event	0

After documenting these types of workflow process templates, follow the instructions under [Designing Workflow Event Types](#) to add any new types of events to the list.

Designing Your External System Feature Configuration

For certain types of external systems, your application may define options used to configure the interaction between your product and your external system. The feature configuration table is used to define some settings applicable to the interaction with your external system, for example:

- The Options grid allows you to define information needed to communicate with the external system. For example, if you interface with an external workforce management system for booking appointments, a feature configuration for the WFM system may use an option to define whether or not the user can manually define an appointment.
- The Messages collection allows you to map each message that you may receive from the external system to a corresponding [message](#) in this system.

NOTE:

Recommendation. We recommend creating a new message for every message that is being mapped. This ensures that changes to future base product messages do not affect the integration.

Define the Message Sender

If any of your messages are routed to an external system via XAI, the message must also be associated with an [XAI Sender](#), which tells the system how to send the message.

NOTE:

Sender Optional. If the NDS message does not require routing, you are not required to define a message sender.

Designing Notif. Download Types

You will need one notification download type for every type of notification your organization can send to service providers. To "design" your notification download types, list:

- Every outgoing notification that you documented under [Designing Workflow Event Types](#).
- Every outgoing notification that you documented under [Designing Workflow Processes To Support Outgoing Notifications](#).
- Every outgoing notification that may be triggered from within the system as described in [System Conditions May Trigger Notification and Workflow](#). For each of these, consider creating a new value for the download type condition flag so that the download type is not hard-coded in the system process that creates the NDS.

We have populated the Notification Download Type column with a few classic examples:

Notification Download Type	Download Type Condition Flag
Inform all parties that customer is stopping service	CUSTSTOP
Inform all parties that meter has been changed	METERCH
Send consumption history	
Check if it's OK to switch customer from current supplier	
Reject request	

You should also consider the priority of each NDS type with respect to the other types.

Once you know the different types of download notifications, you have to determine the physical method used to route the notification to the third party. Refer to [Designing Notification Download Profiles](#) for how to do this.

If the outbound notification may be routed via XAI, you must define an [XAI inbound service](#). This is used by the XAI MPL to retrieve information about the record related to the outbound message.

FASTPATH:

Refer to [Configuring the System for XAI Messages](#) for more information on designing your XAI outgoing messages.

Designing Context Values for the NDS Type

A notification download staging record must reference some "relevant data" that allows the extract process or the XAI MPL process to retrieve the data needed to send to the external system. For example, if the person's name has changed, the NDS must reference the Person Id. When formatting the data to send to an external system, the appropriate process would take the person ID and retrieve the data needed to send to the external system.

The NDS may use either the context collection or the characteristics collection to enter this "relevant data". If you choose to use context, you may decide to define the context types for the NDS type (although it is not required).

If NDS records of this type are interfaced through XAI, then the context types and corresponding XPATH values are required for every piece of data that is required to call an application service to build the extract. The XPATH indicates the relative path in the XML document where the field value will be placed when building the XML document.

FASTPATH:

Refer to [Configuring the System for XAI Messages](#) for more information on designing your XAI outbound messages.

Configuring the System for XAI Messages

The following provides links so that you can easily navigate the information related to configuring the system for XAI messages:

- [Designing Your Real Time Messages](#)
- [Designing the Real Time Message XSL](#)
- [Designing the Real Time Message Route Types](#)
- [Designing the Real Time Message NDS Types and Download Profile](#)
- [Designing the Real Time Message NDS Types and Download Profile](#)
- [Designing the Near Real Time Response](#)

Designing Your Real Time Messages

Consider the following integration scenario. A page in your product is configured to send information to an external system real time. If the connection is down, the user is warned and may authorize that the message is sent in near real time. The following sections describe how to design the control tables needed for both scenarios.

[Top of Page](#)

Designing the Real Time Message XSL

During implementation, you will determine the information required to send to the external system. The real time message engine is passed all the data available in the page model as an XML document. The request XSL must be designed to map this data into a format expected by the target.

In addition, you should create two response XSLs to transform the message received from the target system.

- One XSL is used for real time responses. The XSL transforms the message into a format understood by the page service user exit. Note that any updates to a system business object as a result of the response is the responsibility of the user exit that called the real time message engine.

- One XSL is used for non-real time responses. In this case, if the response should update a system object, the response must trigger an XUS to update the system object. Therefore, the XSL must map the response into an XML request that reference the appropriate service to update the system object.

[Top of Page](#)

Designing the Real Time Message Route Types

For this scenario, you must create two route types.

- Create an [XAI Route Type](#) for the real time message. Indicate the sender, the request XSL and the response XSL used for real time responses.

NOTE:

The sender referenced on the route type should be one that supports synchronous communication (such as an HTTP sender).

- Create an XAI route type for the near real time message. Indicate the sender, the request XSL and the response XSL used for near real time responses. Mark the route type to check the flags Receive Acknowledge and Post Response. These flags indicate to XAI that a response should be received and that additional processing is required.

[Top of Page](#)

Designing the Real Time Message NDS Types and Download Profile

For this scenario you must create two NDS types

- One NDS type is used for the real time message. Indicate an appropriate notification download condition. This allows the mechanism that creates the NDS to refer to this condition rather than hard-coding the NDS type.
- One NDS type is used for the near real time message. This NDS type must reference a special XAI inbound service used to indicate to the download staging sender that the XDS does not need to be created.

NOTE:

Download Condition. The real time message engine receives the NDS type as input. It is the user exit's responsibility to determine the appropriate NDS type based on a notification download condition.

You'll need to create an entry in the service provider's [notification download profile](#) for each NDS type. Set the processing method to XAI and indicate the appropriate route type (real time or near real time) created above.

[Top of Page](#)

Designing the Near Real Time Response

To process a response to the near real time message, you must also design the XAI inbound service to be used to process the response.

- The XSL transformation scripts that are referenced by the inbound service should populate the appropriate XML elements to allow XAI to recognize that the inbound message is a response.
- Otherwise, the inbound service should be designed to process this message as appropriate. For example, if the response should update the status of the record that initiated the outgoing request, the inbound service should reference the appropriate schema to update the appropriate record.

[Top of Page](#)

Designing Near Real Time NDS Messages

As described above, near real time NDS messages require you to define:

- A [download staging receiver](#) to process records on the NDS table.

- A download staging [sender](#) to process "responses" to the download staging receiver.

The following sections identify other control table design issues.

Consider the following integration scenario. A change to a certain type of record in your product should trigger a message to an external system to inform that system of the change. This will be a near real time message and we expect an asynchronous response.

- [Designing the XML Request and Inbound Service](#)
- [Designing the Near Real Time NDS Type](#)
- [Designing the Near Real Time Route Type](#)
- [Designing the Response](#)

Designing the XML Request and Inbound Service

You should work with the external system to determine the information they require for this message. If a system service already exists to extract all the required information, you may use that service. However, it's possible that you would need to create a new service to extract the information you need.

Your implementers must create an appropriate service and then use the [schema editor](#) to create request and response schemas based on this service.

NOTE:

Message ID. If you expect an asynchronous response to this message, the request and response schemas must include a private attribute to hold the MessageID.

Create an XAI inbound service for extracting the appropriate information and building the XML request.

[Top of Page](#)

Designing the Near Real Time NDS Type

Define an appropriate NDS type. The information defined here is required to successfully build the XML request.

- Indicate the XAI Inbound Service created above.
- Indicate an appropriate notification download condition. This allows the mechanism that creates the NDS to refer to this condition rather than hard-coding the NDS type.
- Use the notification download type context to identify the data required to build the request to invoke the XAI Inbound Service for this download type. In our case, context types must be defined for the business object id and the message id. Use the [XPath](#) to indicate the relative page for this element in the request schema.

Context Type	XPATH
Maintenance Object ID	//ExtractInfoService/ExtractInfoHeader@MaintenanceObjectID
Message ID	//ExtractInfoService/ExtractInfoHeader/MessageID

[Top of Page](#)

Designing the Near Real Time Route Type

You must create an [XAI Route Type](#). The route type identifies the sender for routing the message and the XSL transformation script required to transform the request into a format expected by the sender.

If the sender supports synchronous communication, you must check the Receive Acknowledge and Post Response flags to successfully process the response. Our scenario indicated that we expect an asynchronous response so we'll leave these flags unchecked.

NOTE:

Routing Optional. If the message does not require routing, you are not required to define an XAI route type.

[Top of Page](#)

Designing the Response

If you expect a response to the message sent, you must also design the XAI inbound service to be used to process the response.

- The XSL transformation scripts that are referenced by the inbound service should populate the appropriate XML elements to allow XAI to recognize that the inbound message is a response.
- Otherwise, the inbound service should be designed to process this message as appropriate. For example, if the response should update the status of the record that initiated the outgoing request, the inbound service should reference the appropriate schema to update the appropriate record.

[Top of Page](#)

Designing Notif. Download Profiles

A notification download profile controls how the system routes notifications to third parties. You associate a notification download profile with one or more service providers. Whenever a notification is sent to a service provider, the system uses the notification download profile to determine the interface method and the format of notifications.

NOTE:

A notification download profile corresponds with a protocol used to route notifications to service providers. If you electronically route all notifications using the same protocol (for example, all service providers in a given jurisdiction may conform to the same record formatting and routing method), you will have a single notification download profile. If you have to route notifications using different protocols to different providers, you will need one notification download profile per protocol.

Using the notification download profile, you can define which outgoing messages are sent as an XML document via the XAI tool and which are sent in batch format. When a message is designated as being sent via the XAI tool, then you typically define XAI routing information. If a message is designated as being sent via batch, then you must indicate the formatting algorithm used by the extract program.

NOTE:

XAI Routing Information Optional. You may indicate that the message should be routed via XAI, but enter no XAI Routing information. You may do this when you are interfacing with one of your own external systems and where the system may be accessed directly. In this scenario, you are using the XAI functionality to communicate between systems using XML documents, but you don't need the routing logic within XAI. Refer to [An NDS Message That Doesn't Require Routing](#) for more information.

The easiest way to design a notification download profile is to choose a representative service provider and design a notification download profile for it by filling in the following matrix. After you've designed a profile, determine how many other service providers can use it. Then design the next service provider's profile and determine who can reuse it. Repeat this process until all your service providers have a profile. Once the profiles are designed, you're ready to set up the control tables.

Background Process used for batch messages	Notification Download Type	Processing Method	XAI Route Type	Format Method
File Transfer	Inform all parties that customer is stopping service	Batch		Format Stop

Inform all parties that meter has been changed	XAI	Meter Change
Send consumption history	Batch	Format Consumption
Check if it's OK to switch customer from current supplier	XAI	Confirm Switch
Reject request	XAI	Reject

Notice that you define the background process at the profile level, but you indicate for each download type whether it will be processed via batch or via XAI. If at least one of the processing methods is batch, then a background process must be entered, otherwise it's not applicable.

Also note that it is possible to link more than one XAI route type to each notification download type formatting method. However, this is not recommended if you expect an [asynchronous response to the NDS message](#).

After you've designed notification download profiles to cover every protocol, you are ready to set up the control tables.

FASTPATH:

Refer to [Notification Download Background Processes](#).

Setting Up Notification and Workflow Procedures

In the previous sections, [Designing Notification Upload & Workflow Procedures](#) and [Designing Notification Downloads](#), we presented a case study that illustrated a mythical organization's workflow procedures. In this section, we'll explain how to set up the control tables to implement these procedures.

Setting Up Workflow Event Types

Workflow event types control what is done by a given workflow event. Open **Admin > Workflow > Workflow Event Type > Add** to define your workflow event types.

FASTPATH:

Refer to [Designing Workflow Event Types](#) for more information.

Description of Page

Enter a unique **Workflow Event Type** and **Description** for the event type.

Turn on **Allow Manual Completion** if an operator is allowed to change the status of workflow events of this type to Complete.

The **Event Activation Algorithm** is used by the system when it activates a workflow event of this type. Refer to [Executing Workflow Events On Their Trigger Date](#) and [Designing Workflow Event Types](#) for more information. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that is associated with workflow event activation. The system comes supplied with several sample algorithm types that should be used as a sample if you have to write a new algorithm type. Click [here](#) to see the algorithm types available for this system event.

The **Event Failure Algorithm** is used by the system when a workflow event of this type fails. This algorithm need only be defined if events of this type can fail. Refer to [Some Workflow Events May Fail](#) and [Designing Workflow Event Types](#) for more information. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that is associated with workflow event failure. The system comes supplied with several sample algorithm types that should be used as a sample if you have to write a new algorithm type. Click [here](#) to see the algorithm types available for this system event.

The **Wait Process** is the background process responsible for monitoring workflow events of this type that are in the Waiting state. Refer to [Waiting Events And Their Waiting Process](#) for more information.

Where Used

Follow this link to view the tables that reference [CI_WF_EVT_TYPE](#) in the data dictionary schema viewer.

Setting Up Workflow Process Templates

A workflow process template defines the workflow events that will be created when a workflow process is created using a template.

FASTPATH:

Refer to [Designing Workflow Process Templates](#) for more information.

Workflow Process Template - Main

Open **Admin > Workflow > Workflow Process Template > Add** to define your workflow process templates.

Description of Page

Enter a unique **Workflow Process Template** code and **Description** for the workflow process template.

The system needs to know the type of workflow process to create when a notification is uploaded that does not contain a valid External System or Notification Upload Type (these two fields are the ones that control the type of workflow process that's created to process the uploaded notification). If you create workflow process templates and label them with a **Notification Condition** of Unknown Notification ID or Unknown Notification Type, the notification upload process will create a respective workflow process when either of the above conditions are discovered. Note: most utilities create an outgoing notification rejecting the uploaded notification when such conditions transpire. The notification condition field is also available for use by your plug-in algorithms for any purpose where you need to identify a specific workflow process.

NOTE:

The values for this field are customizable using the Lookup table. This field name is WF_PROC_COND_FLG.

Use **Comments**. to describe the workflow process template.

When a workflow process is created, the system links one or more workflow events to it. The information in the **Workflow Responses** scroll defines these events and when they will be triggered. The following fields are required for each event:

Event Sequence. Sequence controls the order in which the workflow events are executed. The sequence number is system-assigned and cannot be changed. If you have to insert a workflow event between two existing events, you'll have to remove the latter events, insert the new event, and then re-specify the removed events.

Workflow Event Type. Specify the type of workflow event to be generated. The event type's description is displayed adjacent.

Dependent on Other Events. Turn this indicator on if the trigger date of the event can only be determined after earlier events are complete. Refer to [Workflow Event Dependencies & Trigger Date](#) for more information. If this switch is on, you must define the events on which this response depends in the **Dependent on Other Events** grid.

Days After Previous Response. Specify the number of calendar days after the completion of the dependent events on which the workflow event will be triggered. If this event is not dependent on the completion of other events, this field contains the number of calendar days after the creation of the workflow process that the related workflow event will be triggered.

When the **Dependent on Other Events** switch is on, a grid appears in which you specify the events on which this event is dependent. The following fields are required for each event:

Sequence. Sequence is system-assigned and cannot be specified or changed.

Dependent on Sequence. Specify the sequence number of the workflow event type on which the above workflow event depends.

Workflow Event Type. The system displays the ID of dependent workflow event in this column.

FASTPATH:

For more information about workflow event templates, see [Setting Up Workflow Event Types](#). For more information about trigger dates, see [Workflow Event Dependencies & Trigger Date](#).

Where Used

A Workflow Process Profile references one or more workflow process templates. Refer to [Setting Up Workflow Process Profiles](#) for more information.

A Workflow Process references a workflow process template. Refer to [Workflow Process - Main](#) for more information.

Workflow Process Template - Template Tree

Open **Admin > Workflow > Workflow Process Template > Search** and navigate to the **Template Tree** page to view information about your workflow process template.

Description of Page

This page is dedicated to a [tree](#) that shows the events linked to the workflow process and information about the event dependencies. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

Setting Up Notification Upload Types

Every notification upload staging record has a notification upload type. This code is one of several fields that control the type of workflow process used to process the incoming notification. Open **Admin > Integration > Notification Upload Type** to define your notification upload types.

FASTPATH:

Refer to [Designing Notification Upload Types](#) for more information.

Description of Page

Enter a recognizable **Notification Upload Type** and **Description**.

Enter a value for the **Upload Condition Flag** when a system condition should trigger the creation of a notification record. Refer to [System Conditions May Trigger Notification and Workflow](#) for more information.

NOTE:

The values for this field are customizable using the Lookup table. This field name is NT_UP_TY_COND_FLG.

If NUS records of this type will be associated with an **Extension** record, indicate the extension here along with its **Navigation Option** to allow a user to navigate to the correct extension record when viewing the notification upload staging record. Refer to [Navigating to Related NUS Extension](#) for more information.

Where Used

A [Notification Upload Staging](#) record must reference a notification upload type.

A [Workflow Process Profile](#) references one or more notification upload types.

Setting Up External Systems

Every notification upload staging record references the system that sent the message. The external system is one of several fields that control the type of workflow process used to process the incoming notification. Refer to [External Systems](#) for more information.

Setting Up Workflow Process Profiles

The system uses a workflow process profile to determine the type of workflow process to create for incoming notifications sent by the service provider. A workflow process profile is associated with one or more service providers. Open **Admin > Workflow > Workflow Process Profile > Add** to define your workflow process profiles.

FASTPATH:

Refer to [Designing Workflow Process Profiles](#) for more information.

Description of Page

Define a unique ID and **Description** for each workflow **Process Profile**.

The information in the grid defines the type of workflow process that will be created for each **Notification Upload Type**. The type of workflow process may differ depending on special criteria. For example, you may have a different workflow process if the customer is industrial (as compared to commercial and residential). You must define Default criteria in case none of the override criteria are met (the Default criteria should have the lowest priority). You MAY have override criteria if different situations result in different types of workflow processes.

The following fields are required for each criterion:

Priority The priority controls the order in which the system determines if the respective workflow process should be used to process notifications of a given type. Higher priorities are checked before lower priorities.

NOTE:

The values for this field are customizable using the Lookup table.

Criteria Algorithm Select the algorithm to be used to check if the workflow process should be initiated for notifications of a given type. If a condition is met, a workflow process is created using the associated workflow process template.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that determines if an incoming notification should be processed using the associated **Workflow Process Template**. The system comes supplied with a sample algorithm type called that

should be used as a sample if you have to write a new algorithm type. Click [here](#) to see the algorithm types available for this system event.

IMPORTANT:

You must have at least one entry in this collection otherwise the system will not start a workflow process when an incoming notification of this type is received. This entry should have the lowest priority code and should reference a **Criteria Algorithm** that references the default workflow criteria algorithm type.

Workflow Process Template Specify the workflow process template to use to process incoming notifications identified with the notification upload type.

FASTPATH:

Refer to [Designing Workflow Process Criteria](#) for more information.

Setting Up Notification Download Types

Every notification download staging record has a notification download type. This code controls the format of the record that is sent to a service provider. Open **Admin > Integration > Notification Download Type** to define your notification download types.

FASTPATH:

Refer to [Designing Notification Download Types](#) for more information.

Notification Download Type - Main

Every notification download staging record has a notification download type. This code controls the format of the record that is sent to a service provider. Open **Admin > Notification Download Type** to define your notification download types.

Description of Page

Enter a unique **Notification Download Type** and **Description**.

Specify the **XAI In Service Name** that will be called for this NDS type if the download profile formatting method indicates that this download type will be processed by XAI.

Enter a value for the **Download Type Condition Flag** when a system condition should trigger the creation of a notification record. Refer to [System Conditions May Trigger Notification and Workflow](#) for more information.

NOTE:

The values for this field are customizable using the Lookup table. This field name is NT_DWN_TY_COND_FLG.

Indicate the relative **Priority** for processing NDS records of this type with respect to other types. This value defaults to Priority 90 - Lowest.

NOTE:

The values for this field are customizable using the Lookup table. This field name is NT_DWN_TY_PRIO_FLG.

Notification Download Type - Context

If the notification download staging record is communicated to the external system through XAI, the NDS context is used to help build the XML document.

Description of Page

The context collection functionality allows you to define a collection of **Context Types** that should be defined for a notification download staging record of this type. When the NDS record is created, with its collection of these Context Types, the Context Values would correspond to system data related to this NDS record. In addition, you may specify a **Context Value** if this is a constant value for NDS records of this type.

The **XPATH** is used by XAI NDS Types. For each context type, use the XPATH to indicate the relative path in the XML document where the field value will be placed when building the XML document.

Where Used

A Notification Download Staging record must reference a notification download type. Refer to [Process X - Populate Notification Upload Staging](#) for more information.

A Notification Download Profile references one or more notification download types. Refer to [Setting Up Notification Download Profiles](#) for more information.

XAI Route Type

Use this page to define information required by the XAI tool to send outgoing messages. Navigate to this page using **Admin > XAI > XAI Route Type > Add**.

Description of Page

Enter the **XAI Route Type**, which defines the information required for outgoing messages, which use the XAI tool. Enter a **Description** for this route type.

The **XSL Request** is the schema used to transform information from the format produced by the system to a format understood by the sender, who receives a message of this type.

The **XSL Response** is the schema used to transform information from the format sent to us by the sender, who responds to this message into a format understood by the system.

Use the **XAISender** to indicate where messages of this type should be sent.

FASTPATH:

Refer to [Message Sender](#) for more information.

Check the **Receive Acknowledge** box if the system expects to receive a synchronous response to outgoing messages of this type.

Check the **Post Response** box if a synchronous response to an outgoing message requires something to occur in the system. If the box is checked, a response to a message of this type causes an [XAI upload staging](#) record to be created. That record is processed along with other uploaded messages, to invoke an appropriate service.

Where Used

All outgoing messages are sent through the notification download staging record. Information related to the formatting of messages is defined on a notification download profile.

Setting Up Notification Download Profiles

A notification download profile controls how the system routes notifications to service providers. You associate a notification download profile with one or more service providers. Whenever a notification is sent to a service provider, the system uses the notification download profile to determine the interface method and the format of notifications.

Open **Admin > Notification Download Profile > Add** to define your notification download profiles.

FASTPATH:

Refer to [Designing Notification Download Profiles](#) for more information.

Description of Page

Define a unique ID and **Description** for each **Download Profile**.

Use **NDS Extract Process** to define the background process that creates notification download records and interfaces them out of the system.

The information in the scroll controls how the **Extract Process** formats an interface record for each **Notification Download Type**. In addition to defining a **Description** and **Comments**, you must define the **Processing Method**. The valid values are XAI and Batch.

If your processing method is Batch, you need to define the **Notification Format Algorithm** that actually formats the interface record. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to [Setting Up Algorithms](#)).
- On this algorithm, reference an Algorithm Type that is associated with workflow event failure. The system comes supplied with several sample algorithm types that should be used as a sample if you have to write a new algorithm type. Click [here](#) to see the algorithm types available for this system event.

If your processing method is XAI, you may define one or more **XAI Route Types**. The [XAI Route Type](#) describes how the message should be formatted and the destination of the message.

Defining Umbrella Agreement Options

Your organization may use umbrella agreement functionality to manage many situations, including (but not limited to) the following:

- Creating a "negotiated contract" with a large customer with many sites.
- Creating a "negotiated contract" with all customers in a common geographic area, for example a specific city or an apartment complex
- Grouping and managing the proposal service agreements created during the quoting process
- Managing the renewal process for special contracts
- Applying override rate terms to one or more service agreements for a given effective period
- more...

NOTE:

Separately module. The umbrella agreement functionality is part of the Contract Management module. If this module is not applicable to your business you may turn it off. Refer to [Turn Off A Function Module](#) for more information.

The Big Picture Of Umbrella Agreements

The topics in this section provide background information about how to configure the system to support your umbrella agreement requirements.

Renewable Umbrella Agreements

When defining your umbrella agreement types, you must determine if umbrella agreements of this type are renewable. If so, you must set the renewal flag on the UA type to Renewable and determine the number of days prior to the end date of the umbrella agreement that the renewal process should start.

In addition, you must decide on your renewal procedure and design an appropriate renewal [algorithm](#). You may use one of the sample renewal algorithms provided by the system.

- The sample algorithm [URNW-TD](#) creates a To Do entry. Using this algorithm you can direct the To Do entry to an appropriate role to determine if the umbrella agreement should be renewed. If the umbrella agreement is associated with an account management group, the To Do entry could be assigned to a role defined for that AMG.
- The sample algorithm [URNW-CA](#) creates a [case](#). Using this algorithm you can create an appropriate case to help track the steps required for renewing an umbrella agreement.

If the base algorithms do not provide the logic your organization needs, you may create a new algorithm.

NOTE:

It is the responsibility of the renewal algorithm to change the renewal date as per the business requirements. If renewal requires manual review, the algorithm should reset the date to blank. If the algorithm can determine a new renewal date, the date should be changed to the appropriate value.

Terms Of Service Covered Entities

For each terms of service type you design, consider whether or not a TOS created for this type should reference one or more entities in the system that are "covered" by the terms of service. Following are some examples of covered entities:

- A special contract is created for all the McDonalds franchises in your service area. When creating the terms of service record for the umbrella agreement used for this contract, you want the user to link the person record representing the McDonalds corporate entity as a covered entity for the TOS.
- An umbrella agreement is created for a large customer with several types of services. Perhaps you want to group the services in the contract based on type of service. The service type may be set up as a covered entity.

The covered entities could also be used to denote eligibility. Refer to [Linking To A UA For A Group Of Premises](#) for an example of using the covered entity collection for eligibility.

The covered entities are actually characteristics. You must define the desired characteristic types if they do not already exist. In this case, they would be foreign key characteristic types. The characteristic types that you want to use must reference a characteristic entity of TOS covered entity.

A terms of service record may only reference covered entity types that you designate on the TOS type record. When designing the covered entities that are allowed for TOS records of a given type, you may also configure the TOS type to indicate that one or more covered entity types are required.

Refer to [Terms Of Service Type - Main](#) for more information.

Overriding Rate Terms

If your organization has business rules that require a service agreement's rate terms to be overridden, you can accomplish this using umbrella agreements and terms of service records. There are two levels of overriding that are possible:

- Override rate schedule
- Override other terms on the rate, including contract riders, contract terms and tax exemptions

To override any rate terms, the service agreement must be linked to a terms of service record that references a template service agreement. In addition, you must configure settings in the system to indicate whether or not overriding of terms is allowed.

- The **SA type** for the customer's service agreement indicates whether or not the rate schedule may be overridden using the rate source flag.
 - If the rate source value is Check TOS First, then SA, when applying the rate, the system determines if the SA is linked to a TOS effective at the time of the bill with a template SA. If so, the rate schedule on the template SA is used.
 - If the rate source value is Check SA Only, when applying the rate, the always uses the rate schedule on the customer's SA (regardless of whether or not the SA is linked to a TOS effective at the time of the bill with a template SA).
- If any bill factor on one of calculation rule indicates that it is eligible for tax exemptions, contract terms or contract riders, the terms of service usage flag on the bill factor indicates where the system should look for the appropriate value. The flag tells the system to find applicable information either on the TOS's template SA only, on the customer's SA only or it should first check the TOS, then check the customer's SA.

The following table illustrates the different scenarios possible when configuring your bill factors and the resulting behavior. The term "standard behavior" in the tables below indicate that the standard behavior as described in [Defining Bill Factors](#) is followed.

	TOS Usage	Template SA	Customer SA	BF Char Source
Contract Rider Applicability is checked	Check TOS First, then SA	Found	Not checked	If source = SA, the template SA's characteristic collection is checked. If source <> SA, the standard behavior is followed.
		Not Found	Found	If source = SA, the customer SA's characteristic collection is checked. If source <> SA, the standard behavior is followed
		Not Found	Not Found	N/A
	Check TOS Only	Found	Not checked	If source = SA, the template SA's characteristic collection is checked. If source <> SA, the standard behavior is followed
		Not Found	Not checked	N/A
		Check SA Only	Not checked	Standard behavior

	TOS Usage	Template SA	Customer SA	BF Value Source	BF Char Source
Value in Contract Term is checked	Check TOS First, then SA	Found	Not checked	Template SA	N/A
		Not Found	Found	Customer SA	N/A

	Not Found	Not Found	Bill Factor Value (based on Char Source)	If source = SA, the template SA's characteristic collection is checked. If source <> SA, the standard behavior is followed
Check TOS Only	Found	Not checked	Template SA	N/A
	Not Found	Not checked	Bill Factor Value (based on Char Source)	If source = SA, the template SA's characteristic collection is checked. If source <> SA, the standard behavior is followed
Check SA Only	Not checked	Standard behavior		
<hr/>				
	TOS Usage	Template SA	Customer SA	BF Value Source
Tax Exemption is checked	Check TOS First, then SA	Found	Not checked	Template SA
		Not Found	Found	Customer SA
		Not Found	Not Found	No value
	Check TOS Only	Found	Not checked	Template SA
		Not Found	Not checked	No value
	Check SA Only	Not checked	Standard behavior	

Note that [proration](#) logic is followed for the above functionality as expected. For example, imagine that a bill factor with Value in Contract Terms is proratable and the TOS usage value is Check TOS First, then SA. If the template SA has a contract value that is only in effect for the first 10 days of a bill, that value is used for the first 10 days and the remaining days in the bill take the value from the SA (if applicable) or the bill factor value as per the table rules above.

NOTE:

When designing your terms of service type, you must indicate whether a template service agreement is optional, required or not allowed. When the template SA is required, the user setting up the TOS record for this TOS type must link the correct service agreement as the template SA. If a new template service agreement is required, the new SA must be created first.

Setting Up Umbrella Agreement Options

The topics in this section describe how to set up umbrella agreement options.

Configure SA Types for Umbrella Agreements

For each type of service agreement that may be linked to a terms of service record, you must configure the Rate Source for the [SA type](#) to indicate if the rate should always be taken from the service agreement (Check SA Only) or if the rate should be taken from the terms of service record for the service agreement, if any (Check TOS First, then SA).

FASTPATH:

Refer to [Overriding Rate Terms](#) for more information.

NOTE:

The rate source is only visible on SA type if the Contract Management module is not [turned off](#).

Configure Bill Factors for Umbrella Agreements

For each bill factor that has a value in contract term, contract rider applicability or tax exemption applicability, you must indicate where the system should look for a value when a service agreement is linked to a terms of service with a template SA and where this bill factor is referenced by the rate being applied. Using the terms of service usage flag, tell the system to Check TOS Only, Check SA Only or Check TOS First, then SA.

FASTPATH:

Refer to [Overriding Rate Terms](#) for more information.

NOTE:

The terms of service usage flag is only visible on bill factor if the Contract Management module is not [turned off](#).

Configure A Campaign To Link An SA To A Terms Of Service

The system provides a pair of column reference algorithm types that support linking a new service agreement to an existing terms of service record via an [order](#). The pair includes a validation algorithm type [CRVL-SATOS](#) and a posting algorithm type [CRPS-SATOS](#).

In order to take advantage of this functionality, you must:

- Create an algorithm for each of the above algorithm types. Only one algorithm is needed for each algorithm type because there are no parameters for either one.
- Setup a unique [column reference](#) for the terms of service ID column reference code and plug in the new validation and posting algorithms on the record.
- For campaigns whose orders link a new service agreement to an existing umbrella agreement, make sure to define a [miscellaneous field](#) for the terms of service ID. Note, the miscellaneous field must reference the column reference that you setup above.

Based on your business practice, you may want to further configure your campaign / package for linking a service agreement to an existing terms of service. Refer to [Linking A Service Agreement To A TOS Through Orders](#) for more information.

Setting Up Umbrella Agreement Types

The umbrella agreement type transaction is used to maintain your UA types. The topics in this section describe how to use this transaction.

Umbrella Agreement Type - Main

Open **Admin > Umbrella Agreement > Umbrella Agreement Type > Add** to define basic information about an umbrella agreement type.

Description of Page

Enter a unique **Umbrella Agreement Type** code and **Description** for the UA type.

Use **Renewal** to indicate whether an umbrella agreement of this type is Renewable or Not Renewable. For renewable UA types, indicate the **Renewal Days Before Expiration**. The renewal date for umbrella agreements of this type are calculated based on this number of days before the end date.

Use the characteristics collection to define characteristics that can be defined for umbrella agreements of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on umbrella agreements of a given type. Turn on the **Default** switch to default the **Characteristic Type** when umbrella agreements of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

NOTE:

Characteristic Types. You can only choose characteristic types defined as permissible on an umbrella agreement record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Enter the valid **Terms of Service Types** that may be referenced for umbrella agreements of this type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_UA_TYPE](#).

Umbrella Agreement Type - Algorithms

Open **Admin > Umbrella Agreement > Umbrella Agreement Type > Search** and navigate to the **Algorithms** page.

Description of Page

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (descriptions of all possible events are provided below).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** for which you can define algorithms.

System Event	Description
UA Renewal	Algorithms of this type are executed by the UARENEW background process when an umbrella agreement has reached its renewal date. Only one renewal algorithm is allowed for a given UA type. Click here to see the algorithm types available for this system event.

Setting Up Terms Of Service Types

The terms of service type transaction is used to maintain your TOS types. The topics in this section describe how to use this transaction.

Terms Of Service Type - Main

Open **Admin > Terms of Service Type > Add** to define basic information about terms of service types.

Description of Page

Enter a unique **Terms of Service Type** code and **Description** for the TOS type.

Use **Template SA Usage** to indicate whether a template SA is Optional, Required or Not Allowed. Refer to [Overriding Rate Terms](#) for more information about using template SAs.

Use the characteristics collection to define characteristics that can be defined for terms of service records of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on terms of service records of a given type. Turn on the **Default** switch to default the **Characteristic Type** when terms of service records of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

NOTE:

Characteristic Types. You can only choose characteristic types defined as permissible on a terms of service record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Use the **Covered Entities** collection to define the types of entities that are "covered" by terms of service records of this type. Refer to [Terms Of Service Covered Entities](#) for more information

NOTE:

Characteristic Types. You can only choose characteristic types defined as permissible for TOS covered entity. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TOS_TYPE](#).

Terms Of Service Type - UA Types

Open **Admin > Umbrella Agreement > Terms of Service Type > Add** and navigate to the **UA Types** page.

Description of Page

Enter the valid **Umbrella Agreement Types** that may reference this terms of service type.

Setting Up TOS Cancel Reasons

Whenever a terms of service record is canceled, a cancel reason must be specified. Open **Admin > TOS Cancel Reason** to specify such cancellation reasons.

Description of Page

Enter an easily recognizable **TOS Cancel Reason** and **Description** for the terms of service cancellation reason.

Umbrella Agreements Configuration Examples

This section provides examples of how you would configure the system to support various business scenarios.

Linking A Service Agreement To A TOS Through Orders

As described in [Configure A Campaign To Link An SA To A Terms Of Service](#), the system has provided a pair of column reference algorithms to enable your users to link a new service agreement to an existing terms of service record (and its umbrella agreement) when starting service via the order transaction.

When designing your campaign and packages to use this option, consider the scenarios where you may use this to determine whether or not you require additional configuration. The following topics describe some possible scenarios.

Linking To A UA For A Given Account

Imagine that you would use this logic when an important customer is starting additional service and has an existing umbrella agreement for other services that the new service should be added to. The user creating the order is probably the account manager for that account and would know the correct terms of service record to link the new service to.

For this scenario, you do not need to provide any "help" to the user to indicate that linking to a TOS is an option or to help select the correct TOS. You simply need to ensure that the campaign that would be used by this account manager to start service for the important customer simply includes a question / miscellaneous field to [link the SA to the terms of service](#).

Linking To A UA For A Group Of Premises

Imagine you have created an umbrella agreement for a group of premises that are not actively managed by a specific user. For example, perhaps there are several apartment complexes that have negotiated a special rate with your company. When new customers sign up for service at a premise that belongs to one of these apartments, the user should be informed that there is a special contract for this premise. The user may also need help selecting the correct terms of service record. To accomplish this logic, you must define additional algorithms to help your users.

For example, perhaps your standard campaign for residential electric customers includes a package to be used by premises linked to one of these apartment buildings and should only be eligible to one of those premises. You must create an eligibility algorithm for this package. Its logic could be designed based on the following assumptions:

- The qualifying premises are linked to a parent premise
- The parent premise is linked to the appropriate terms of service record in its covered entity collection

The eligibility algorithm checks to see if the order's premise is linked to a parent premise that is referenced on a terms of service record for an umbrella agreement that is currently in effect. It could also be restricted to search for a specific umbrella agreement type or terms of service type if appropriate.

NOTE:

The campaign for this scenario must include a question / miscellaneous field to [link the SA to the terms of service](#). It should be configured with an applicability of only applicable on package. The package that is used to include the SA into the terms of service for the apartment complex would include this question / miscellaneous field.

Once the user chooses this package, user is prompted to provide the correct terms of service record via the question / miscellaneous field. To help the user, you should create a column reference retrieval algorithm to default the correct terms of service record for this premise. It would use the same logic as the eligibility algorithm described above. In other words, it would find the parent premise for this premise and then find the terms of service record covered entity collection includes this parent premise and whose umbrella agreement is currently in effect.

Reports Addendum

This chapter is an addendum to the general [Defining and Designing Reports](#) chapter.

Description of Sample Reports

This section provides an overview of each sample report supplied with Oracle Utilities Customer Care and Billing that may be found in the demonstration database. They may be used by your organization as they are or as a starting point for creating a [new report](#).

NOTE:

Account Security. The sample reports provided with the product do NOT incorporate account security. If a user has been given security to view the report, then all the data in the report is available for viewing.

Active Severance Processes - CI_ACSVPR

Parameters

Parameter	Parameter Code	Description
Start Date	P_FROM_DT	Start date to use for reporting severance processes. If not defined, start date is set to 1900-01-01.
End Date	P_TO_DT	End date to use for reporting active severance processes. If not defined to date is set to the current date.

Report Description

This report is used to aid in the monitoring of active severance processes. The report details are used to review how the company is doing as far as the collection of outstanding debt, as well as to monitor the progress of currently active severance processes.

The report selects active severance processes whose creation date is between the input start and end dates.

Bill Print in BI Publisher - CI_BILLPR

Parameters

Parameter	Parameter Code	Description
CC&B user	P_USER_ID	No default value. Required. Returns an error message if user ID is null, empty or does not exist in database.
Batch Switch	P_BATCH_SW	For a single bill print, Batch Switch is 'No' and for multiple bill print, Batch Switch is 'Yes'.
First Bill	P_FROM_BILL_ID	First Bill Id to print for the Bill Range.
Last Bill	P_TO_BILL_ID	Last Bill Id to print for the Bill Range.
Batch Code	P_BATCH_CD	Passed in by the system the report is submitted in batch. Only bills with a Bill Routing that references this batch code are selected.
Batch Number	P_BATCH_NBR	Passed in by the system the report is submitted in batch. Only bills with a Bill Routing that references this batch number are selected.
Extract Algorithm	P_EXTRACT_ALG	Passed in by the system the report is submitted in batch. Only bills with a Bill Routing whose bill route type references this extract algorithm are selected.

Report Description

This report is used with BI Publisher to display all the information that appears on the customer's printed bill for the bill ID entered in the input parameters. Refer to the description of the parameters above. The bill is only displayed if the bill is complete.

See below for details of the information extracted for the report. In addition, the sample report layout provided for the base product includes:

- Payment coupon.
- Form for change of address.
- A graph of consumption for current charges.

The following sections highlight the data that is extracted for this report.

Bill Information

Field Name	Format	Source/Value/Description
SORT_KEY	A12	Primary sort for the report - currently sorted by Bill Id
BILL_ID	A12	CI_BILL
SEQNO	N5	CI_BILL_ROUTING
PER_ID	A10	CI_BILL_ROUTING
NBR_BILL_COPIES	N1	CI_BILL_ROUTING
ENTITY_NAME1	A64	CI_BILL_ROUTING. Mailing Entity Name 1.

ENTITY_NAME2	A64	CI_BILL_ROUTING. Mailing Entity Name 2.
ENTITY_NAME3	A64	CI_BILL_ROUTING. Mailing Entity Name 3.
COUNTRY	A3	CI_BILL_ROUTING
ADDRESS1	A254	CI_BILL_ROUTING
ADDRESS2	A254	CI_BILL_ROUTING
ADDRESS3	A254	CI_BILL_ROUTING
ADDRESS4	A254	CI_BILL_ROUTING
CITY	A30	CI_BILL_ROUTING
COUNTY	A30	CI_BILL_ROUTING
STATE	A6	CI_BILL_ROUTING
POSTAL	A12	CI_BILL_ROUTING
IN_CITY_LIMIT	A1	CI_BILL_ROUTING
BATCH_CD	A8	CI_BILL_ROUTING
BATCH_NBR	N10	CI_BILL_ROUTING
NO_BATCH_PRT_SW	A1	CI_BILL_ROUTING
ACCT_ID	A10	CI_BILL
BILL_STAT_FLG	A2	CI_BILL
BILL_DT	Date	CI_BILL
DUE_DT	Date	CI_BILL
ACCT_ENTITY_NAME	A64	Get Main Person from CI_PER_NAME. Account Entity Name
LANGUAGE_CD	A3	Person's Language from CI_PER
CURRENCY_CD	A3	CI_ACCT
CUR_SYMBOL	A4	CI_CURRENCY_CD
DECIMAL_POSITIONS	N1	CI_CURRENCY_CD
CUR_POS_FLG	A2	CI_CURRENCY_CD
OPEN_ITEM_SW	A1	CI_CUST_CL
BILL_COPY	A1	Bill copy number

Bill Segment Information

Field Name	Format	Source/Value/Description
BSEG_ID	A12	CI_BSEG
SA_ID	A10	CI_BSEG
START_DT	Date	CI_BSEG
END_DT	Date	CI_BSEG
BILL_CYC_CD	A4	CI_BSEG
WIN_START_DT	Date	CI_BSEG

EST_SW	A1	CI_BSEG
CLOSING_BSEG_SW	A1	CI_BSEG
PREM_ID	A10	CI_BSEG
CIS_DIVISION	A5	CI_SA
SA_TYPE_CD	A8	CI_SA
SAT_DESCR	A60	CI_SA_TYPE_L
BILL_PRT_PRIO_FLG	A2	CI_SA_TYPE
GRAPH_UOM_CD	A4	CI_SA_TYPE
HEADER_SEQ	N3	CI_BSEG_CALC
BCALC_START_DT	Date	CI_BSEG_CALC
BCALC_END_DT	Date	CI_BSEG_CALC
RS_CD	A8	CI_BSEG_CALC
EFFDT	Date	CI_BSEG_CALC
BILLABLE_CHG_ID	A12	CI_BSEG_CALC
CALC_AMT	N15.2	CI_BSEG_CALC
BCALC_DESC_BILL	A80	CI_BSEG_CALC
SEQNO	N5	CI_BSEG_CALC_LN
PRT_SW	A1	CI_BSEG_CALC_LN
APP_IN_SUMM_SW	A1	CI_BSEG_CALC_LN
CALC_LN_AMT	N15.2	CI_BSEG_CALC_LN
DST_ID	A10	CI_BSEG_CALC_LN
MSR_PEAK_QTY_SW	A1	CI_BSEG_CALC_LN
DESCR_LN	A80	CI_BSEG_CALC_LN
STATE	A6	CI_PREM
CITY	A30	CI_PREM
POSTAL	A12	CI_PREM
SIBLING_ID	A12	CI_FT

Account Summary Information

Field Name	Format	Source/Value/Description
CURRENCY_CD	A3	Derived from CI_BILL_SA or CI_FT.
SUM_CUR_AMT	S15.2	Get Bill's Current Balance from CI_BILL_SA Bill's Current Charge from CI_FT Bill's Current Correction Charge from CI_FT Bill's Current Adjustment from CI_FT Bill's Current Payment from CI_FT
SUM_TOT_AMT	S15.2	Get Bill's Total Balance from CI_BILL_SA

Bill's Total Charge from CI_FT
 Bill's Total Correction Charge from CI_FT
 Bill's Total Adjustment from CI_FT
 Bill's Total Payment from CI_FT

BILL_PRT_FLG	A4	Set as follows: ASBL - Indicator for Bill's Balance from CI_BILL_SA ASBS - Indicator for Bill's Current Charge from CI_FT ASBC - Indicator for Bill's Correction from CI_FT ASAD - Indicator for Bill's Adjustment from CI_FT ASPS - Indicator for Bill's Payment from CI_FT
--------------	----	---

Current Charge Information

Field Name	Format	Source/Value/Description
SUM_CUR_AMT	S15.2	Derived from CI_FT (Only the current charge bill segments.)
SUM_TOT_AMT	S15.2	Derived from CI_FT (Only the current charge bill segments.)
CURRENCY_CD	A3	CI_BILL_SA or CI_FT
DEBT_CL_CD	A4	Debt Class of SA Type
DESCR	A60	Debt Class Description

Premise Information

One record for every premise linked to the bill.

Field Name	Format	Source/Value/Description
PREM_ID	A10	Premise Id obtained from Bill Segment
PREM_TYPE_CD	A8	CI_PREM
LL_ID	A10	CI_PREM
MAIL_ADDR_SW	A1	CI_PREM
TREND_AREA_CD,	A8	CI_PREM
COUNTRY	A3	CI_PREM
ADDRESS1	A254	CI_PREM
ADDRESS2	A254	CI_PREM

ADDRESS3	A254	CI_PREM
ADDRESS4	A254	CI_PREM
CITY	A30	CI_PREM
NUM1	A6	CI_PREM
NUM2	A4	CI_PREM
HOUSE_TYPE	A2	CI_PREM
COUNTY	A30	CI_PREM
STATE	A6	CI_PREM
POSTAL	A12	CI_PREM
GEO_CODE	A11	CI_PREM
IN_CITY_LIMIT	A1	CI_PREM

Auto Pay Information

Field Name	Format	Source/Value/Description
BILL_ID	A12	CI_APAY_CLR_STG
APAY_SRC_CD	A12	CI_APAY_CLR_STG
EXT_ACCT_ID	A50	CI_APAY_CLR_STG
SCHED_EXTRACT_DT	Date	CI_APAY_CLR_STG. The date that the automatic payment will be downloaded.
APAY_SRC_NAME	A30	CI_APAY_SRC_L
APAY_SRC_DESCR	A60	CI_APAY_SRC_L
TNDR_TYPE_DESCR	A60	CI_TENDER_TYPE_L
TENDER_TYPE_CD	A4	CI_TENDER_TYPE

Bill Segment Readings

Field Name	Format	Source/Value/Description
BSEG_ID	A12	CI_BSEG_READ
SP_ID	A10	CI_BSEG_READ
SEQNO	N5	CI_BSEG_READ
REG_CONST	S12.6	CI_BSEG_READ
USAGE_FLG	A2	CI_BSEG_READ
USE_PCT	S3	CI_BSEG_READ
HOW_TO_USE_FLG	A2	CI_BSEG_READ
MSR_PEAK_QTY_SW	A1	CI_BSEG_READ
UOM_CD	A4	CI_BSEG_READ

TOU_CD	A8	CI_BSEG_READ
SQI_CD	A8	CI_BSEG_READ
START_REG_READ_ID	A12	CI_BSEG_READ
START_READ_DTTM	DTTM	CI_BSEG_READ
START_REG_READING	S15.6	CI_BSEG_READ
END_REG_READ_ID	A12	CI_BSEG_READ
END_READ_DTTM	DTTM	CI_BSEG_READ
END_REG_READING	S15.6	CI_BSEG_READ
MSR_QTY	S18.6	CI_BSEG_READ
FINAL_UOM_CD	A4	CI_BSEG_READ
FINAL_TOU_CD	A8	CI_BSEG_READ
FINAL_SQI_CD	A8	CI_BSEG_READ
FINAL_REG_QTY	S18.6	CI_BSEG_READ
BADGE_NBR	A30	CI_MTR
NBR_OF_DGTS_RGT	S2	CI_REG

Bill Segment Consumption History

Field Name	Format	Source/Value/Description
SA_ID	A10	CI_BSEG
START_DT	Date	CI_BSEG
BSEG_ID	A12	CI_BSEG
END_DT	Date	CI_BSEG
UOM_CD	A4	CI_BSEG_SQ
TOU_CD	A8	CI_BSEG_SQ
SQI_CD	A8	CI_BSEG_SQ
QTY	S18.6	Service Quantity from CI_BSEG_SQ

Payment Information

One record for every payment linked to the bill.

Field Name	Format	Source/Value/Description
PAY_DT	Date	CI_PAY_EVENT
FT_TYPE_FLG	A2	Pay or Pay Cancel ('PS' or 'PX')
CAN_RSN_CD	A4	CI_PAY
CURRENCY_CD	A3	CI_FT.

CUR_AMT	S15.2	Derived from CI_FT. Sum of CUR_AMT for the bill with FT Type Flag = (PS or PX) and Show on Bill Switch = 'Yes'
TOT_AMT	S15.2	Derived from CI_FT. Sum of CUR_AMT for the bill with FT Type Flag = (PS or PX) and Show on Bill Switch = 'Yes'
TOT_PREV_BAL	S15.2	Calculated as Previous Balance = Ending Balance - (current charges + payments + adjustments + corrections)
PAY_CAN_RSN_DESCR	A60	Description of cancel reason code. (Note, that this is retrieved in a special stored procedure that retrieves all the cancel reason codes and descriptions.)

Adjustment Information

One record for every adjustment linked to the bill.

Field Name	Format	Source/Value/Description
BILL_ID	A12	CI_FT
ACCOUNTING_DT	Date	CI_FT
ARS_DT	Date	CI_FT
PARENT_ID	A12	CI_FT
SIBLING_ID	A12	CI_FT
FT_ID	A12	CI_FT
CUR_AMT	S15.2	CI_FT
TOT_AMT	S15.2	CI_FT
FT_TYPE_FLG	A2	Adjustment or Adjustment Cancel ('AD' or 'AX')
SHOW_ON_BILL_SW	A1	CI_FT
CURRENCY_CD	A3	CI_FT
XFERRED_OUT_SW	A1	CI_FT
ADJ_TYPE_CD	A8	CI_ADJ
CAN_RSN_CD	A4	CI_ADJ
SA_ID	A10	CI_ADJ
CHAR_PREM_ID	A10	CI_SA
DESCR_ON_BILL	A254	CI_ADJ_TYPE_L
ADJ_CAN_RSN_DESCR	A60	Description of cancel reason code. (Note that this is retrieved in a special stored procedure that retrieves all the cancel reason codes and descriptions.)

Bill Message Information

One record for every bill message linked to the bill.

Field Name	Format	Source/Value/Description
MSG_PRIORITY_FLG	A12	CI_BILL_SA or CI_FT
INSERT_CD	S15.2	CI_BILL_SA or CI_FT
MSG_ON_BILL	S15.2	CI_BILL_SA or CI_FT

Billed Revenues by Rate - CI_BILREV

Parameters

Parameter	Parameter Code	Description
Accounting Period	P_ACCT_PERIOD	Defines the accounting period used for the report. A valid fiscal year and accounting period for a valid accounting calendar must be provided.
Account Type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an Account Type.
Account Type	P_REV_ACCTY_CHAR	Account Type Char Value for Revenue related GL Accounts. The char type defined for this parameter should match the Char Type code defined as parameter #2. The parameter value indicated for this parameter should be one that represents revenue accounts.

Report Description

This is an analysis report for the billed revenues for an accounting period according to the various rates, which were in effect in the system. The information in this report helps to adjust rates in order to achieve better financial results and comply with regulations and market trends.

This report selects all records in the financial transaction GL collection that satisfy the following criteria:

- The financial transaction is frozen.
- The Accounting Date on the financial transaction within input accounting period (parameter 1)
- The distribution code associated with the GL entry has a characteristic type and value that matches the input Account Type Characteristic and Account Type (parameters 2 & 3)

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow

of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Case Statistics By Case Type - CI_CSESTS

Parameters

Parameter	Parameter Code	Description
Start Date	P_FROM_DT	See the report's description for how this field is used. If start date is not specified, it is defaulted to 7 days prior to the end date.
End Date	P_TO_DT	See the report's description for how this field is used. If end date is not specified, it is defaulted to the current processing date.
Case Condition (Open, Closed)	P_COND_FLG	If specified, only cases in this condition are included in the report. If left blank, the reports produces statistics for both open and closed cases.

Report Description

This report provides two types of statistics:

1. Open cases whose creation date falls between the input Start Date and End Date (inclusive)
2. Closed cases whose closing date falls between the input Start Date and End Date (inclusive)

The third parameter is only used if you want to restrict the statistics to only open or closed cases. If you leave this parameter blank, both open and closed statistics will be produced.

The following information is provided in graphical format:

- Number of cases by case type
- Percentage of cases by case type

Case Statistics for a Given Status - CI_CSESGS

Parameters

Parameter	Parameter Code	Description
Start Date	P_FROM_DT	See the report's description for how this field is used. If start date is not specified, it is defaulted to 7 days prior to the end date.
End Date	P_TO_DT	See the report's description for how this field is used. If end date is not specified, it is defaulted to the current processing date.

Case Type/Status	P_CASE_STATUS_CD	This is the desired Case Type and Status that will be reported on.
Responsible User	P_CASE_OWNER	If specified, only cases with this responsible user are included in the report.
First Bucket High Limit	P_B1_LIMIT	Cases that took <= this number of days to reach the given status will be grouped together for statistical reporting.
Second Bucket High Limit	P_B2_LIMIT	Cases that took <= this number of days but more than the first bucket high limit to reach the given status will be grouped together for statistics reporting.
Third Bucket High Limit	P_B3_LIMIT	Cases that took <= this number of days but more than the second bucket high limit to reach the given status will be grouped together for statistics reporting. Cases that took more than this number of days are included in another group.

Report Description

This report shows cases of a given case type that transitioned to a given status during a given date range.

Graphs are printed to show the number and percentage of cases grouped by the time it took to reach the status. These statistics are grouped into age buckets whose size is controlled by the last 3 parameters.

Summary statistics are also printed showing the minimum, maximum, average and median times for these cases.

Collection Summary - CI_CLLSUM

Parameters

Parameter	Parameter Code	Description
Start Date	P_FROM_DT	If from date is not set, the default is one month prior to the current date.
End Date	P_TO_DT	End date of the date range. If not defined by user, it is set to one month after the current date

Report Description

This report provides detailed monthly summary information of all collection activities. The report is typically used by a collection department for resource planning and performance review purposes.

This report selects pending and completed collection events whose event trigger date falls between the input start and end dates.

Customer Contacts by Type - CI_CUSTCN

Parameters

Parameter	Parameter Code	Description
-----------	----------------	-------------

Start Date	P_FROM_DT	Start date to use for reporting customer contacts. If not defined, the start date is set to the current date minus 7 days.
End Date	P_TO_DT	End date to use for reporting customer contacts. If not defined, End Date is set to the current date.
Customer Contact Class / Type	P_CC_TYPE_CD	Specify a Customer Contact Class / Type to restrict the report output to a specific class / type.

Report Description

This report lists all customer contacts in the system created within the input date range. You may optionally restrict the report to customer contacts for a given Customer Contact Class / Type (parameter 3).

NOTE:

Graphs. The information on this report is shown in both textual and graphical formats.

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_CC table on CC_DTTM to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Customers with Life-Support / Sensitive-Load - CI_PMLSSL

Parameters

Parameter	Parameter Code	Description
Service Type	P_SERVICE_TYPE	Service type used to restrict the report to premises with services of this type.

Report Description

This reports displays a detailed list of premises that are coded with Life-Support (LS) or Sensitive-Load (SL) information. It may optionally restrict the output to premises with service points for input service type.

This information is used by a company to make sure that customers at these premises are dealt with appropriately in the case of an outage (planned and unplanned) or service cut due to non-payment.

The report provides detailed information about the premise and the related facility elements providing service to that premise (e.g. substation, feeder and node for electric service).

NOTE:

If ANY person connected to an account has an LS/SL indication in the Person record, then all the premises connected to this account will be treated as LS/SL. This means, for example, that a premise connected to an account that has a 3rd party guarantor with LS/SL is treated as a premise with LS/SL.

Field Order Print in BI Publisher- CI_FOPRNT

NOTE:

Field Order Print in batch for BI Publisher is not supported yet.

Parameters

Parameter	Parameter Code	Description
CC&B user	P_USER_ID	No default value. Required. Returns an error message if user ID is null, empty or does not exist in database.
Batch Switch	P_BATCH_SW	For a single FO print, Batch Switch is 'No' and for multiple FO print, Batch Switch is 'Yes'.
First FO	P_FROM_FO_ID	First FO ID to print for the FO Range.
Last FO	P_TO_FO_ID	Last FO ID to print for the FO Range.
Batch Code	P_BATCH_CD	Batch Code passed in by the system the report is submitted in batch. Only field orders that reference this batch code are selected.
Batch Number	P_BATCH_NBR	Batch Code passed in by the system the report is submitted in batch. Only field orders that reference this batch number are selected.
Extract Algorithm	P_EXTRACT_ALG	Passed in by the system the report is submitted in batch. Only field orders whose dispatch groups reference this FO extract algorithm are selected.

Report Description

This report is used with BI Publisher to display the Field Order information for the field order ID entered in the input parameters. Refer to the description of the parameters above. The field order will only be displayed if the field order's status is dispatched.

The following sections highlight the data that is extracted for this report.

Field Order Information

Field Name	Format	Source/Value/Description
LANGUAGE_CD	A3	SC_USER
SORT_KEY	A10	FO Id
FO_ID	A10	CI_FO
PREM_ID	A10	CI_FO
FO_SCHED_DTTM	Date	CI_FO.SCHED_DTTM
FO_STATUS_FLG	A2	CI_FO
WORK_DTTM	Date	CI_FO
DISP_GRP_CD	A8	CI_FO
REP_CD	A8	CI_FO
WORKED_BY	A8	CI_FO

EXTRACT_NEXT_SW	A1	CI_FO
EXTRACT_DTTM	Date	CI_FO
FO_DESCR	A254	CI_FO.DESCR254
BATCH_CD	A8	CI_FO_STG_DWN
BATCH_NBR	N10	CI_FO_STG_DWN
FA_ID	A10	CI_FA
SP_ID	A10	CI_FA
FA_TYPE_CD	A8	CI_FA
FA_PRIORITY_FLG	A2	CI_FA
FA_CREATED_BY_FLG	A2	CI_FA
FA_SCHED_DTTM	Date	CI_FA.SCHED_DTTM
FA_STATUS_FLG	A2	CI_FA
ELIG_DISPATCH_SW	A1	CI_FA
CRE_DTTM	Date	CI_FA
INSTRUCTIONS	A254	CI_FA
TEST_SEL_ID	A10	CI_FA
FA_DESCR	A254	CI_FA.DESCR254
FA_CAN_RSN_CD	A8	CI_FA_STEP
STEP_SEQ_NBR	N3	CI_FA_STEP
FA_STEP_TY_ACT_FLG	A2	CI_FA_STEP
STP_ENTITY_FLG	A4	CI_FA_STEP
SP_MTR_HIST_ID	A10	CI_FA_STEP
SP_ITEM_HIST_ID	A10	CI_FA_STEP
MR_ID	A12	CI_FA_STEP
MTR_CFG_MTR_ID	A10	CI_FA_STEP
MTR_ID	A10	CI_FA_STEP
ITEM_ID	A10	CI_FA_STEP
CC_ID	A10	CI_FA_STEP
SPAWNED_FA_ID	A10	CI_FA_STEP
ACCT_ID	A10	CI_FA_STEP
DV_TEST_ID	A10	CI_FA_STEP
PREM_TYPE_CD	A8	CI_PREM
LL_ID	A10	CI_PREM
MAIL_ADDR_SW	A1	CI_PREM
KEY_SW	A1	CI_PREM
KEY_ID	A10	CI_PREM
OK_TO_ENTER_SW	A1	CI_PREM
MR_INSTR_CD	A4	CI_PREM
MR_INSTR_DETAILS	A250	CI_PREM

MR_WARN_CD	A4	CI_PREM
TREND_AREA_CD	A8	CI_PREM
COUNTRY	A3	CI_PREM
ADDRESS1	A254	CI_PREM
ADDRESS2	A254	CI_PREM
ADDRESS3	A254	CI_PREM
ADDRESS4	A254	CI_PREM
CITY	A30	CI_PREM
NUM1	A6	CI_PREM
NUM2	A5	CI_PREM
HOUSE_TYPE	A2	CI_PREM
COUNTY	A30	CI_PREM
STATE	A6	CI_PREM
POSTAL	A12	CI_PREM
GEO_CODE	A11	CI_PREM
IN_CITY_LIMIT	A1	CI_PREM
FA_STEP_TY_DESCR	A60	CI_LOOKUP.DESCR / FA Step Type Action Flag
OPTIONAL_SW	A1	CI_FA_STEP_TYPE
FA_STEP_TYPE_DESCR	A60	CI_FA_STEP_TYPE_L.DESCR
REP_DESCR	A60	CI_REP_L.DESCR
FA_TYPE_DESCR	A60	CI_FA_TYPE_L.DESCR
DISP_GRP_DESCR	A60	CI_DISP_GRP_L.DESCR
LL_ACCT_ID	A10	CI_LANDLORD.ACCT_ID
LL_DESCR	A60	CI_LANDLORD_L.DESCR
MR_WARN_DESCR	A60	CI_MR_WARN_L.DESCR
MR_INSTR_DESCR	A60	CI_MR_INSTR_L.DESCR

Account / Person Information

Field Name	Format	Source/Value/Description
PREM_ID	A10	CI_FO
ACCT_ID	A10	CI_SA
MAIN_CUST_SW	A1	CI_ACCT_PER
PER_ID	A10	CI_ACCT_PER
ACCT_REL_TYPE_CD	A8	CI_ACCT_PER
ENTITY_NAME	A64	CI_PER_NAME
LS_SL_FLG	A2	CI_PER

Person Phone Information

Field Name	Format	Source/Value/Description
PER_ID	A10	CI_ACCT_PER - Primary person
SEQ_NUM	N3	CI_PER_PHONE
PHONE_TYPE_CD	A12	CI_PER_PHONE
COUNTRY_CODE	A3	CI_PER_PHONE
PHONE	A24	CI_PER_PHONE
EXTENSION	A6	CI_PER_PHONE
DESCR	A60	CI_PHONE_TYPE_L

Premise Geo Information

Field Name	Format	Source/Value/Description
PREM_ID	A10	CI_FO
GEO_TYPE_CD	A8	CI_PREM_GEO
GEO_VAL	A50	CI_PREM_GEO
GEO_TYPE_DESCR	A60	CI_GEO_TYPE_L.DESCR

SP Type Information

Field Name	Format	Source/Value/Description
SP_ID	A10	CI_FA
SP_DESCR	A254	CI_SP.DESCR254
SP_TYPE_CD	A8	CI_SP
SP_TYPE_DESCR	A60	CI_SP_TYPE_L.DESCR
SP_SUBTYPE_FLG	A2	CI_SP_TYPE
SP_SUBTYPE_DESCR	A60	CI_LOOKUP/SP Subtype Flag
SP_STATUS_FLG	A2	CI_SP
INSTALL_DT	Date	CI_SP
SP_SRC_STATUS_FLG	A2	CI_SP
DISCON_LOC_CD	A4	CI_SP
DISCON_LOC_DESCR	A60	CI_DISCON_LOC_L

MR_CYC_CD	A4	CI_SP
MR_CYC_DESCR	A60	CI_MR_CYC_L
MTR_LOC_CD	A4	CI_SP
MTR_LOC_DESCR	A60	CI_MTR_LOC_L
MTR_LOC_DETAILS	A250	CI_SP
FAC_LVL_1_CD	A8	CI_SP
FAC_LVL_1_DESCR	A60	CI_FAC_LVL_1_L.DESCR
FAC_LVL_2_CD	A8	CI_SP
FAC_LVL_2_DESCR	A60	CI_FAC_LVL_3_L.DESCR
FAC_LVL_3_CD	A8	CI_SP
FAC_LVL_3_DESCR	A60	CI_FAC_LVL_3_L.DESCR

SP Installed Meter Information

Field Name	Format	Source/Value/Description
SP_ID	A10	CI_FA
SP_MTR_HIST_ID	A10	CI_SP_MTR_HIST
MTR_CONFIG_ID	A10	CI_SP_MTR_HIST
EFF_DTTM	Date	CI_MTR_CONFIG
MTR_ID	A10	CI_MTR_CONFIG
INSTALL_DTTM	Date	CI_MR.READ_DTTM
READ_DTTM	Date	CI_MR
BADGE_NBR	A30	CI_MTR
MTR_TYPE_CD	A8	CI_MTR
MTR_STATUS_FLG	A2	CI_MTR
MFG_CD	A8	CI_MTR
MODEL_CD	A8	CI_MTR
SERIAL_NBR	A16	CI_MTR
RECEIVE_DT	Date	CI_MTR
MTR_DESCR	A254	CI_MTR.DESCR254
REG_ID	A10	CI_REG
READ_SEQ	N2	CI_REG
UOM_CD	A4	CI_REG
TOU_CD	A8	CI_REG
REG_CONST	N12.6	CI_REG
CONSUM_SUB_FLG	A2	CI_REG
HOW_TO_USE_FLG	A2	CI_REG
NBR_OF_DGTS_LFT	N2	CI_REG

NBR_OF_DGTS_RGT	N2	CI_REG
FULL_SCALE	N18.7	CI_REG
READ_OUT_TYPE_CD	A8	CI_REG
PROTOCOL_CD	A8	CI_REG
TOLERANCE	N14.5	CI_REG
REG_READ_ID	A12	CI_REG_READ
MR_ID	A12	CI_REG_READ
READ_TYPE_FLG	A2	CI_REG_READ
REG_READING	N15.6	CI_REG_READ
MTR_TYPE_DESCR	A60	CI_MTR_TYPE_L.DESCR
MFG_DESCR	A60	CI_MFG_L.DESCR
MODEL_DESCR	A60	CI_MODEL_L.DESCR

SP Installed Item Information

Field Name	Format	Source/Value/Description
SP_ID	A10	CI_FA
SP_ITEM_HIST_ID	A10	CI_SP_ITEM_HIST
ITEM_ID	A10	CI_SP_ITEM_HIST
INSTALL_DTTM	Date	CI_SP_ITEM_EVT.EVENT_DTTM
BADGE_NBR	A30	CI_ITEM
ITEM_TYPE_CD	A8	CI_ITEM
ITEM_STATUS_FLG	A2	CI_ITEM
MFG_CD	A8	CI_ITEM
MODEL_CD	A8	CI_ITEM
SERIAL_NBR	A16	CI_ITEM
RECEIVE_DT	Date	CI_ITEM
ITEM_DESCR	A254	CI_ITEM.DESCR254
ITEM_TYPE_DESCR	A60	CI_ITEM_TYPE_L.DESCR
MFG_DESCR	A60	CI_MFG_L.DESCR
MODEL_DESCR	A60	CI_MODEL_L.DESCR

SP Geo Information

Field Name	Format	Source/Value/Description
SP_ID	A10	CI_FA
GEO_TYPE_CD	A8	CI_SP_GEO

GEO_VAL	A50	CI_SP_GEO
GEO_TYPE_DESCR	A60	CI_GEO_TYPE_L.DESCR

SP Characteristics Information

Field Name	Format	Source/Value/Description
SP_ID	A10	CI_FA
CHAR_TYPE_CD	A8	CI_SP_CHAR
EFFDT	Date	CI_SP_CHAR
CHAR_VAL	A16	CI_SP_CHAR
ADHOC_CHAR_VAL	A254	CI_SP_CHAR
CHAR_VAL_FK1	A50	CI_SP_CHAR
CHAR_VAL_FK2	A50	CI_SP_CHAR
CHAR_VAL_FK3	A50	CI_SP_CHAR
CHAR_VAL_FK4	A50	CI_SP_CHAR
CHAR_VAL_FK5	A50	CI_SP_CHAR
CHAR_TYPE_FLG	A4	CI_CHAR_TYPE
FK_REF_CD	A8	CI_CHAR_TYPE
CHAR_TYPE_DESCR	A60	CI_CHAR_TYPE_L.DESCR

SP Multi Item Information

Field Name	Format	Source/Value/Description
SP_ID	A10	CI_FA
EFFDT	Date	CI_MULT_ITEM
ITEM_TYPE_CD	A8	CI_MULT_ITEM
ITEM_CNT	N11,2	CI_MULT_ITEM
ITEM_TYPE_DESCR	A60	CI_ITEM_TYPE_L.DESCR

FA Severance Process Information

Field Name	Format	Source/Value/Description
FA_ID	A10	CI_FA
SEV_PROC_ID	A10	CI_SEV_PROC
SA_ID	A10	CI_SEV_PROC

SEV_ARS_DT	Date	CI_SEV_PROC
ACCT_ID	A10	CI_SA

GL Accounting Summary - CI_GLACSM

Parameters

Parameter	Parameter Code	Description
Accounting Period	P_ACCT_PERIOD	Defines the accounting period used for the report. A valid fiscal year and accounting period for a valid accounting calendar must be provided.
Account Type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an Account Type. The account types for the GL accounts are used for grouping the output to the report.

Report Description

This is a financial audit report used to check the financial details in Oracle Utilities Customer Care and Billing for an accounting period against the GL system. The report summarizes all financial transaction (FT) information for a given accounting period according to the different operating and GL divisions and according to various levels of the account GL information.

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Letter Print - BI Publisher Sample Welcome Letters - CI_LTRGN_ENG REPORT

NOTE:

Generating letters in batch is not supported yet.

Parameters

Parameter	Parameter Code	Description
Batch Switch	P_BATCH_SW	For reports accessed online, set the switch to 'No'. Otherwise, it should be set to 'Yes'.

Report Description

This sample report template for BI Publisher produces letters that are not associated with any other object (i.e., the template does not have to extract information from another object to merge into a letter). You can use this template as a welcome letter for a new customer.

By default, the English versions of the report templates are provided with the base product. If multilingual report templates are required, your implementation should provide reports for each language. When a letter is generated, the system uses the report template based on the customer's language.

The address and name for the company are extracted from the [installation options](#). The text for the letter is defined in the report layout and not provided by Oracle Utilities Customer Care and Billing. Reports are printed according to the customer's language definition and not based on the user's language definition.

This sample report uses the following text:

Welcome to %1. You have been filed with ID Number %2.

We hope to provide you with our best possible service. If you experience any problems or have any questions, please contact one of our customer service representatives at (800)1234567.

%1 is the company name stored as a message on the [installation options](#).

%2 is Person Id stored on the Customer Contact.

Meter Reads Performance - CI_MTREAD

Parameters

Parameter	Parameter Code	Description
Accounting Period	P_ACCT_PERIOD	Defines the accounting period used for the report. A valid fiscal year and accounting period for a valid accounting calendar must be provided.

Report Description

This report selects bill segments for the input accounting period and displays the total number of read meters and unread meters for these bill segments grouped by route type.

Meters that are considered read are meters whose register reads have a status of Regular or Verified . Meters that are considered unread are meters whose register reads have a status of System Estimate, Office Estimate, System Prorate or Billing Force.

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Open Cases By Type - CI_CSEOPN

Parameters

Parameter	Parameter Code	Description
Start Date	P_FROM_DT	See the report's description for how this field is used. If start date is not specified, it is defaulted to 7 days prior to the end date.
End Date	P_TO_DT	See the report's description for how this field is used. If end date is not specified, it is defaulted to the current processing date.
Case Type	P_CASE_TYPE_CD	If specified, only cases of this type are included in the report.
Responsible User	P_CASE_OWNER	If specified, only cases with this responsible user are included in the report.
First Bucket High Limit	P_B1_LIMIT	Cases that are open for less than or equal to this number of days will be grouped together for statistical reporting.
Second Bucket High Limit	P_B2_LIMIT	Cases that are open less than or equal to this number of days but more than the first bucket high limit will be grouped together for statistics reporting.
Third Bucket High Limit	P_B3_LIMIT	Cases that are open less than or equal to this number of days but more than the second bucket high limit will be grouped together for statistics reporting. Cases that took more than this number of days are included in another group.

Report Description

This is a report on open cases that were created between a given date range.

The report can be limited to a specific type and/or responsible user.

For each case type, the report shows the following:

- Number of open cases by age bucket (the last 3 parameters control the size (in days) of each bucket)
- Percentage of open cases by age bucket
- Details of the open cases

Payments Balance - CI_PMTBAL

Parameters

Parameter	Parameter Code	Description
-----------	----------------	-------------

Start Date	P_FROM_DT	Report gets all the payments that have been received during a given date range (from and to date parameters). If start date is not defined by the user, it is set to 7 days prior to the current date.
End Date	P_TO_DT	End date of the date range. If not defined by user it is set to the current date

Report Description

This report provides an overall view of all payments created within the input date range. It is typically used for financial control and audit purposes. The report provides summary information about valid payments received and about canceled payment. Data is summarized by the tender source and the type of payment.

Receivables Aging - CI_RCVAGA

Parameters

Parameter	Parameter Code	Description
Cutoff Date	P_CUTOFF_DATE	The date from which the arrears buckets are calculated. If no value is entered, the default is the current date minus 7 days.
First Bucket High Limit	P_B1_LIMIT	High limit of first bucket.
Second Bucket High Limit	P_B2_LIMIT	High limit of second bucket.
Third Bucket High Limit	P_B3_LIMIT	High limit of third bucket.

Report Description

This report lists all accounts and their arrears information as of the input cutoff date using a balance forward accounting method.

Outstanding debt is placed into the buckets provided as input using the age of the debt as of the cutoff date. Credits are applied to the oldest debt first. For each account a separate bucket is used to display new charges. In addition, the total accounts receivable balance is displayed for each account.

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ARS_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Tax Payables Analysis - CI_TXPYBL

Parameters

Parameter	Parameter Code	Description
Start Date	P_FROM_DT	Show summary of the tax amounts starting from this date. If not specified, the system will

		default this value to the current date minus 7 days.
End Date	P_TO_DT	Show summary of the tax amounts up to this date. If not specified, the system will default this value to the current date.
Account Type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an Account Type.
Account Type	P_TAX_ACCTY_CHAR	Account Type Char Value for tax related GL Accounts. The char type defined for this parameter should match the Char Type code defined as parameter #3. The parameter value indicated for this parameter should be one that represents tax liability accounts.

Report Description

This report displays a summary of the tax amounts that were levied by the company to customers within the input date range. It also includes the tax exemption information for that period.

This report select all records in the financial transaction GL collection that satisfy the following criteria:

- The financial transaction is frozen.
- The Accounting Date on the financial transaction within the input date range
- The distribution code associated with the GL entry has a characteristic type and value that matches the input Account Type Characteristic and Account Type (parameters 3 & 4)

The report also provides tax exemption information for bill segments whose financial transactions satisfy the above criteria. The tax exemption information is retrieved by looking at the bill calculation lines associated with the FT's bill segment.

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

To Do Entries - CI_TDENTR

Parameters

Parameter	Parameter Code	Description
ToDo Entry Status	P_ENTRY_STATUS_FLG	Defines if the To Do entries on the report should be limited to those with a given status value. If this parameter is left blank, the report will show all open and being worked To Do entries.
ToDo Type	P_TD_TYPE_CD	Defines if the To Do entries on the report should be limited to those of a given ToDo

type. If this parameter is left blank, the report will show all ToDo type that have at least one open or being worked entry.

Report Description

The report shows open and being worked To Do entries.

You can limit the report to entries in a given status by specifying the desired **To Do Status** (open or being worked). If you don't specify a status, all open and being worked To Do entries will appear on this report.

You can also limit the report to entries of a given To Do Type by specifying the desired To Do Type. If you don't specify a To Do Type, all To Do Types with at least one entry in the open / being worked state will appear on this report.

NOTE:

Graphs. The information on this report is shown in both textual and graphical formats.

Umbrella Agreement Summary - CI_UASUMM

Parameters

At least one parameter is required. If the umbrella agreement is entered, no other parameter value is allowed. The report definition uses the validation algorithm [RPTV-UASUMM](#) to check these conditions.

Parameter	Parameter Code	Description
Umbrella Agreement	P_UA_ID	Specify the Umbrella Agreement to restrict the report to this umbrella agreement (regardless of its status). When this parameter is specified, all other parameters are not allowed.
Account Management Group	P_AMG	Specify a value for this parameter to restrict the report to umbrella agreements related to this Account Management Group.
Umbrella Agreement Characteristic Type	P_CHAR_TYPE	Specify a value for this parameter to restrict the report to completed umbrella agreements related to this Characteristic type and value (parameter 4).
Umbrella Agreement Characteristic Value	P_CHAR_VALUE	Specify a value for this parameter to restrict the report to completed umbrella agreements related to this Characteristic type (parameter 3) and value.
Number of Days Before Expiration	P_EXPIRE_IN_X_DAYS	Specify a value for this parameter to restrict the report to completed umbrella agreements whose End Date is on or before the current date less the Number of Days before Expiration.

Report Description

This report provides summary information about one or more [umbrella agreements](#).

If the user does not supply an umbrella agreement ID, but enters one or more of the other parameters, the report only selects umbrella agreements that match the input criteria AND the match the following criteria:

- The status of the umbrella agreement is complete
- The start and end dates of the umbrella agreement include the current date

The report does not include canceled terms of service records or canceled service agreements linked to the above umbrella agreements.

For each umbrella agreement that is included in the report, the output includes information about the umbrella agreement, its collection of terms of service records and for each TOS record, its collection of service agreements.

In addition, the report includes the following graphs:

- Umbrella agreement summary graph showing billing history by accounting period for the effective dates of the umbrella agreement for all service agreements linked to the umbrella agreement's terms of service records.
- Terms of service summary graph showing billing history by accounting period for the effective dates of the umbrella agreement for all its service agreements

Unbilled Revenues - CI_UBLREV

Parameters

Parameter	Parameter Code	Description
Accounting Period	P_ACCT_PERIOD	Defines the accounting period used for the report. A valid fiscal year and accounting period for a valid accounting calendar must be provided.
Account Type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an Account Type.
Account Type	P_REV_ACCTY_CHAR	Account Type Char Value for Revenue related GL Accounts. The char type defined for this parameter should match the Char Type code defined as parameter #2. The parameter value indicated for this parameter should be one that represents revenue accounts.

Report Description

This report provides a simplified calculation of estimated unbilled revenue for a given month.

It processes frozen financial transactions associated with bill segments and canceled bill segments whose bill segment end date is within the **Accounting Period** (parameter 1). It determines the billed revenue for the FT and then uses this information to calculate the unbilled revenue.

To determine "revenue", the report summarizes amounts posted to any distribution code with a characteristic entry that matches the input **Account Type Characteristic** type and **Account Type** characteristic value (parameters 2 and 3).

The estimate for the unbilled portion is calculated as (Bill Amount / Number of Days in the Bill Period) * Number of Unbilled Days for the Accounting Period.

For example, consider a bill segment on a monthly-billed cycle for the period of 3/10/03 - 4/9/03, and for \$150 (revenue part of the bill). Assume we are in the April/03 accounting period, which covers 4/1/03 through 4/30/03. This means 21 days are unbilled for April. The unbilled revenue is calculated as $150/31 * 21 = \$101.61$.

The following are some points to note about this report.

- If the report runs for an historical date, it still estimates the unbilled revenue portion based on the above formula even if actual data exists. In other words, it will not try to find actual bills for subsequent months. As a result, this report always shows what would have been the estimated unbilled revenue for a particular month if that month is the most recent one.
- Services can be billed monthly bi-monthly, quarterly or in any desired frequency. This report only performs the revenue recognition estimation using actual bills that were created in the report's accounting period. As a result, it does not estimate unbilled revenue for accounts that were not billed in any portion of the input accounting period.
- Revenue recognition practices are unique and may vary from customer to customer. In a given month we can produce current bills (e.g. for electric service) or future bills (e.g. cable services). In addition in some cases we can have bills that started before the accounting period and ending after it (for example in a cancel-rebill situation). This report will only estimate unbilled revenues for bills with the accounting period equals to the report's accounting period.

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_BSEG table on START_DT, END_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Vacant Premises with Consumption - CI_VACANT

Report Description

This report shows all premises that are considered vacant, and provides information about the level of consumption, the period of vacancy, and the service point and register information.

This report selects service points that are linked to a service agreement that is either canceled or closed. The report excludes any service points that have never been linked to a service agreement. If consumption has been registered at such a service point since the end date of the service agreement, the service point's details are displayed.

This report may be used by the utility to investigate sites where problems such as service theft or leakage may be occurring.

NOTE:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_SA_SP on STOP_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. In addition, many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any effect on the production environment.

Defining Overdue Processing Options

The system periodically monitors how much your customers owe to ensure they haven't violated your collection rules. When a violation is detected, the system initiates the appropriate activities (e.g., letters, disconnect notices, collection agency referrals, and eventually write off). The topics in this section describe how to configure the system to manage your overdue processing requirements.

NOTE:

The overdue processing module has been designed to collect on virtually anything from an unpaid bill to an unmatched financial transaction. You tell the system what you collect on by configuring the various overdue processing control tables. In this release, the base-package is delivered with algorithms that support collecting on overdue bills. If your

organization practices balance-forward accounting (i.e., collection is based on overdue service agreement balances), you will not use this functionality. Rather, you will use the functionality described under [Defining Credit and Collections Options](#).

WARNING:

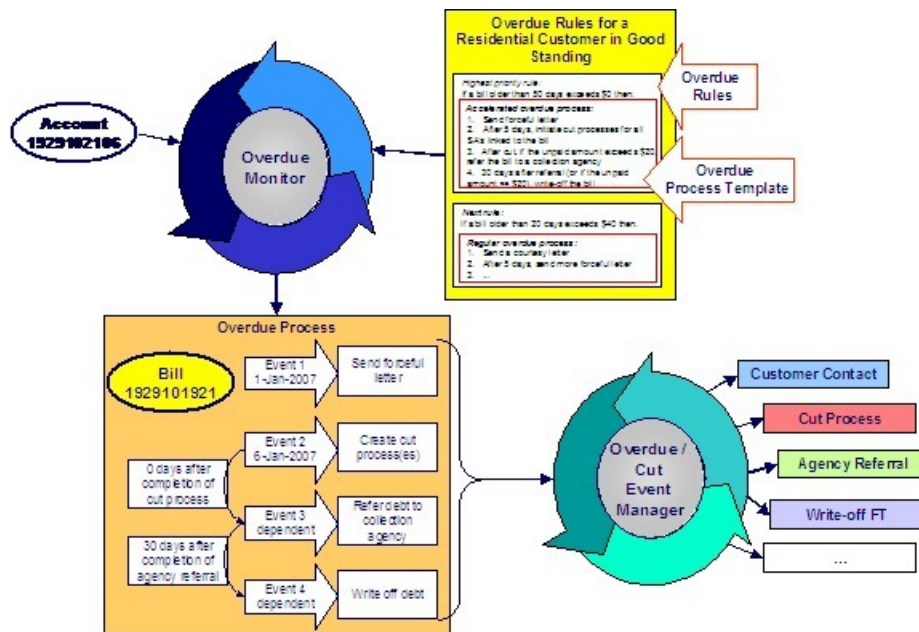
Setting up this module is as challenging as your organization's collection rules. If you have simple rules, the set up process will be straightforward. If your rules are complicated (e.g., they differ based on the type of customer, the age of debt, the type of service, etc.), your setup process will be more challenging.

Case Study - Collecting On Overdue Bills

The following topics introduce a case study that describes how overdue processing can be used to collect on overdue bills. This is just an example as virtually every aspect of overdue processing is configurable. Use this case study to familiarize yourself with the overdue processing core concepts.

Monitoring Overdue Bills

The following diagram illustrates the objects and processes involved with collecting overdue bills.



There are many important concepts illustrated above:

The **Overdue Monitor** checks if your accounts have bills that violate your overdue rules

The **Overdue Monitor** is a background process that periodically reviews your account's financial obligations.

Note well: every account belongs to a collection class. There are two types of collection classes: those whose accounts are monitored by the **Account Debt Monitor**, and those that are monitored by the **Overdue Monitor**. This chapter describes the **Overdue Monitor**.

Overdue rules define when and how unpaid bills are collected

An account's [collection class overdue rules](#) have algorithms that monitor an account's financial obligations. These algorithms are invoked by the Overdue Monitor when it's [time to review](#) an account's obligations.

These algorithms can contain any type of criteria. However, most are defined using a combination of a threshold age and monetary amount. For example, a classic algorithm would check if a bill has unpaid financial transactions more than 20 days old that exceed \$50.

In the case of bill-oriented collection, the monitoring algorithms look at each of the account's bills to determine if they are paid. Note, a bill is considered paid if its financial transactions (FTs) are linked to a balanced match event. If a monitoring algorithm finds an unpaid bill, it can check if it old enough (and large enough) to be considered a violation.

When you set up a monitoring algorithm, you define the type of overdue process that should be created when an overdue bill is detected. You do this by defining the appropriate "overdue process template".

An overdue process template defines how to handle an overdue bill

An [overdue process template](#) contains one or more [overdue event types](#). These define the number and type of events that are created to prod the customer to pay. For example, you might set up an overdue process template with event types to send a series of letters followed up by a call.

The overdue process template has contains the rules defining [when events are activated](#).

The specific action that's performed by an overdue event is controlled by the Activation algorithm defined on its event type. Refer to [Overdue Event Type - Main](#) for a list of the various Activation algorithms delivered with the base package.

Multiple objects can be associated with a single process

The above diagram shows a single bill linked to an overdue process. It should be noted that an overdue process is capable of referencing multiple bills (or other objects).

Note well: while a single overdue process can reference many overdue objects, all such objects must be of the same type. For example, you cannot commingle bills and service agreements under a single overdue process. The type of object managed by an overdue process is defined on its [overdue process template](#).

If a customer pays the bill, the overdue process is cancelled

If an overdue bill is paid, the [overdue process is canceled](#) real-time. You control if and how an overdue process is cancelled by setting up the appropriate rules on the [overdue process template](#).

The Overdue / Cut Event Manager activates and triggers overdue events

The [Overdue / Cut Event Manager](#) is a background process that activates overdue events on the appropriate date. On this date, the event's Activation algorithm(s) are called.

This Overdue / Cut Event Manager also has the responsibility of recursively activating later events that are dependent on the completion of earlier events.

Events can be activated real-time

[Overdue Process - Main](#) has a button that allows users to activate (and recursively trigger) overdue events online / real-time. This means you don't have to wait for a batch job to activate events.

Overdue events can wait for related activities to complete

As described above, an overdue event's Activation algorithm can create virtually any object. What wasn't explained is that the event can be set up to [wait](#) for the ancillary object to finish before it completes. For example, an event can create a To Do entry and wait for it to complete before the next event is triggered. You can introduce plug-ins to create and wait on virtually any object.

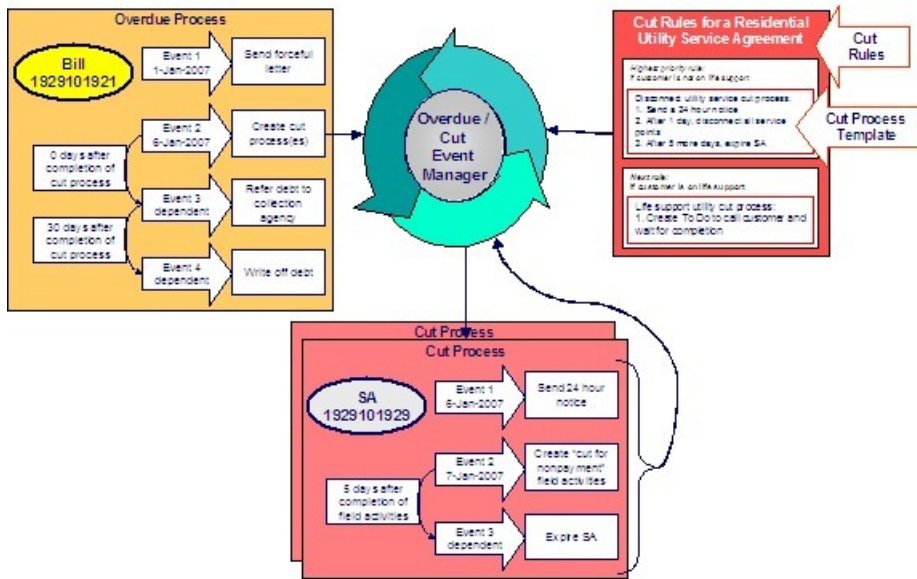
While an overdue event is in the Wait state, the Overdue / Cut Event Manager monitors the state of the related object(s). When the related object completes, the event is transitioned to the Complete state (thus triggering dependent overdue events). Please see [Some Events Wait For Something Before Completing](#) for more information.

Cutting Service Agreements

An overdue process may contain an overdue event that creates a cut process(es). A cut process is used to cut (i.e., stop) a service agreement. The following diagram illustrates the objects and processes involved with cutting a service agreement.

NOTE:

Cut processes aren't required. It's quite conceivable to design an overdue process that doesn't cut service. For example, the overdue process may just contain an event that creates a case and the case manages the collection activities.



An overdue process may contain an overdue event that cuts (i.e., stops) service

If an overdue process's bill remains unpaid, one of the latter overdue events typically creates one or more [cut processes](#). A cut process contains the activities to stop a service agreement (in the hopes that lack of service will inspire the customer to pay). It should be noted that a separate cut process is created for each service agreement.

Cut rules define how to sever service agreements

The system allows you to define rules to control the type of cut process created. For example, you may have a different cut process if the customer has life support equipment, or if it's winter, or ...

Overdue events can "wait" for related activities to complete	<p>The cut rules are defined on the service agreement's SA type. This allows different rules for different types of service agreements.</p>
A cut process template defines how to cut a service agreement	<p>When an overdue event creates cut process(es), it typically enters the Wait state. It enters this state as it is waiting for the cut process(es) to complete before it can itself complete. While in the Wait state, the Overdue / Cut Event Manager monitors the state of the related cut process(es). When the cut process(es) complete, the system transitions the originating overdue event to the Complete state (thus triggering dependent events).</p> <p>Cut processes and events are created using a cut process template. A cut process template defines the actions involved in cutting a given type of service agreement. A cut process template usually contains several cut events. These events are a series of letters and / or disconnection field activities that eventually result in the expiration of a service agreement if payment is not received.</p> <p>An Activation algorithm controls the specific action associated with a cut event and therefore an event can do practically anything. Refer to Cut Event Type - Main for a list of the various Activation algorithms delivered with the base package.</p>
Cut events are also managed by the Overdue / Cut Event Manager	<p>The same background process that manages the activation of overdue events manages cut events. This means you only have to submit one batch job to activate and trigger both overdue and cut events.</p>
Events can be activated real-time	<p>Cut Process - Main has a button that allows users to activate (and recursively trigger) cut events online / real-time. This means you don't have to wait for a batch job to activate events.</p>
Cut events can "wait" just like overdue events	<p>Cut events support the ability to wait for something to complete just like overdue events. For example, if a cut event creates a field activity, it enters the Wait state. While in the Wait state, the Overdue / Cut Event Manager monitors the state of the related field activities. When the field activities complete, the system transitions the originating cut event to the Complete state.</p> <p>And, just like for overdue events, the notion of a cut event creating something and then waiting for it to complete is not limited to field activities. For example, you could have a cut event create a To Do entry and wait for it to complete before the next event is triggered.</p> <p>Please see Some Events Wait For Something Before Completing for more information.</p>
After a service agreement is stopped, it will be final billed	<p>Many cut processes are configured so their last cut event expires the service agreement. What happens at expiration depends on the type of service agreement. However, eventually the service agreement is transitioned into the Stopped state.</p> <p>When the last active service agreement linked to an account is stopped, the system changes the account's bill cycle to bill that evening. If only one of many SAs is stopped, the SA will only be final billed as per the account's original bill cycle schedule.</p>
If a customer doesn't pay their final bill, the final bill will be processed using a different type of overdue process	<p>Final bills only differ from ongoing bills in that the service agreements associated with the bill's FTs are not active. This means that there is nothing to cut. This means that the type of overdue process used to handle a "final" bill should differ from that used to handle "ongoing" bills.</p>

As described above, overdue rules define how unpaid bills are handled (both ongoing and final). We recommend configuring your overdue rules to use different overdue process templates for final versus ongoing bills.

The last overdue event typically causes the debt to be written off

Ultimately, if the overdue and cut events fail to inspire payment, the debt will be written off. The method used to write-off a bill is controlled by an overdue event Activation algorithm.

How Does The Overdue Monitor Work?

This section describes how the Overdue Monitor background process (batch control: C1-ADMOV) uses your overdue rules to collect overdue debt.

NOTE:

Recommendation. We recommend that you familiarize yourself with the concepts described in the [case studies](#) before reading this section.

Collecting overdue bills. The examples in the following section frequently refer to how the Overdue Monitor is configured for an organization that collects on unpaid bills. It should be noted that these are just examples as the Overdue Monitor can be used to collect on virtually anything (if you create the appropriate plug-in algorithms).

Different Overdue Rules For Different Customers

The Overdue Monitor uses rules to control how it monitors an account's debt. The system allows you to define different rules for different combinations of collection class, division and currency code. For example,

- You probably have different collection rules for different jurisdictions (i.e., CIS Divisions). For example, if you have customers in different states / provinces, you may have different rules applied to each jurisdiction's debt. The **CIS division on each customer's account** defines the jurisdictional rules applied to the account's debt.
- You probably have different collection rules for different customer segments. For example, bills for large customers might be overdue if they are more than 10 days late, whereas small customers might have 24 days. You differentiate your customers in respect of the overdue via the **collection class on the customers' accounts**. An account's initial collection class is defaulted from its customer class. You may override an account's collection class at will.
- You will have different criteria for every currency in which you work because the monitoring process always compares a customer's debt against some value and this value must be denominated in the customer's currency. A customer's currency is defined using a **currency code on the account**.

The above means that every combination of CIS division, collection class, and currency code can have different rules. The following matrix is used to illustrate a sample organization's rules (note, we assume a single currency and therefore avoid the third dimension):

Account's Division	Account's Collection Class:	Account's Collection Class:
	Commercial Customer	Residential Customer
North	Highest Priority: If a bill exists with unpaid FT's > \$0 that is older than 45 days, create the commercial 45 days late overdue process.	Highest Priority: If a bill exists with unpaid FT's > \$0 that is older than 50 days, create the accelerated overdue process for residential customers.

Next Priority: If a bill exists with unpaid FT's > \$100 that is older than 30 days, create the commercial 30 days late overdue process.

Lower Priority: If a bill exists with unpaid FT's > \$25 that is older than 25 days, create the courtesy reminder overdue process for residential customers.

South

Notice that there can be multiple criteria for each cell in the matrix. What differentiates one criteria from another is its priority. The higher priority criteria will be compared first. If the debt violates the criteria, the overdue process is initiated and no further comparisons are performed.

Overdue Rules Are Embodied In Algorithms

Your organization's overdue rules are defined in algorithms plugged in on [Collection Class Overdue Rules](#) (in the Overdue Monitor Rule system event). When the Overdue Monitor analyzes an account, it uses the rules associated with the account's collection class, CIS division and currency code. To analyze an account, it simply invokes these algorithms in sequence order, i.e., the lower the sequence, the higher its priority.

An Overdue Monitor Rule algorithm has two responsibilities:

- it determines if an account violates its overdue rules,
- if so, it creates one or more overdue processes using an [overdue process template](#)

When Is An Account Monitored?

The Overdue Monitor determines if an account violates your overdue rules at least every X days, where X is defined on the [Customer Class - Controls](#) associated with the account's customer class and division (in the field Min Credit Review Freq (Days)).

In addition, the Overdue Monitor examines an account's debt as follows:

- X days after an account's bill due date (X is defined in the field Credit Review Grace Days on [customer class control](#)).
- If a payment is canceled with a cancellation reason that indicates non-sufficient funds.
- If a payment arrangement is broken (assuming you use the base package's break payment arrangement plug-in). Refer to [Breaking A Bill Oriented Payment Arrangement](#) for more information.
- If a pay plan is broken. Refer to [The Pay Plan Monitor](#) for more information.
- If a [match event](#) is added, changed or deleted.
- When a written off bill is [reversed](#).

NOTE:

Additional events. Your implementation can have other events trigger the analysis of an account by the Overdue Monitor. To do this, add logic to insert a row on the CI_ADM_RVW_SCH table when the event occurs. This row simply references the account ID to be reviewed and the desired review date.

Collection Class Defines If And How Accounts Are Monitored

As described above, every account references a collection class. The collection class defines if and how its accounts are monitored. There are three options:

- The accounts are monitored by the [Account Debt Monitor](#)
 - The accounts are monitored by the Overdue Monitor (this is described in this chapter).
 - The accounts are not monitored for overdue debt.
-

NOTE:

Migration. If you plan to migrate from the Account Debt Monitor (ADM) method of collection to the Overdue Monitor method, a special Overdue Monitor algorithm ([CI-ACT-CSW](#)) has been supplied that will skip accounts that are eligible for review by the Overdue Monitor if they have an active collection, severance or write-off process. This algorithm should be plugged in the applicable Collection Class Overdue Rules so that it will be invoked first. This allows accounts with active ADM-oriented collection activities to work themselves through the system. When an account no longer has any active ADM-oriented activities, monitoring responsibilities will be assumed by the Overdue Monitor.

The Big Picture Of Overdue Processes

As described above, the Overdue Monitor subjects your accounts to overdue rules. If a rule is violated, an overdue process is created. The topics in this section provide background information about overdue processes.

How Are Overdue Processes Created?

As described [above](#), the system creates an overdue process when an account violates your overdue rules. In addition, a user can manually create an overdue process at their discretion.

The Components Of An Overdue Process

The following topics describe the major components of an overdue process.

Overdue Objects

When an overdue process is created, the system links the overdue object(s) to the process. For example, if an overdue bill is detected, the bill is linked to the overdue process.

When you set up an [overdue process template](#), you define the type of object it collects on by defining the [foreign key characteristic type](#) used to reference the object. For example, when you set up an overdue process template to collect on bills, you define a foreign key characteristic type that references the bill object.

Overdue Events

An overdue process has one or more overdue events. These events are the actions designed to encourage the customer to pay. For example, you might set up overdue events that:

- Send letters (via the creation of a customer contact)
- Create To Do entries
- Impact the account's credit rating

- Create a case (e.g., to seize the customer's assets)
- ... (the list is only limited by your imagination as algorithms are used to perform the event's actions)

You define the number and type of events by configuring [overdue process templates](#). When the system creates an overdue process, it copies the events defined on the specified template.

It's important to note that all overdue events are created when the overdue process is created. A separate background process, the [Overdue / Cut Event Manager](#), is responsible for activating, monitoring, and triggering overdue events. Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do entry to a user, start a cut process, refer debt to a collection agency, write-off debt, etc.).

Overdue Log

Every overdue process has a log holding its history. Entries are added to the log when meaningful events occur, for example:

- When the process is created, a log entry is created to describe why the process was started.
- When an overdue event is activated, a log entry is created. These entries frequently contain a foreign key to the object that the event created so that users can easily drill down to the object from the log. For example, if an event creates a To Do entry, the To Do entry's foreign key is placed on the log and this allows a user to drill down on the log entry to see the To Do entry.
- When a process is canceled, a log entry is created to describe the circumstances of the cancellation (e.g., manual versus automated).
- Users can manually add log entries (you might want to think of these as "diary" entries) as desired.
- ...

Many of the base-package algorithms involved in overdue processing insert log entries so that a thorough audit trail is maintained. These algorithms have been designed to allow you to control the verbiage in each log entry by defining the desired message number using an algorithm parameter.

The log is viewable on the [Overdue Process - Log](#) page.

NOTE:

More than just an audit trail. The log entries are more than just an audit trail. The system makes use of the log entries to know what it did. For example, when an overdue event needs to monitor the state of the To Do entries that it created, it uses the log to determine the identity of these To Do entries.

Experimenting With Alternative Overdue Process Templates

The system allows you to determine the efficacy of proposed overdue process templates using a small subset of customers before implementing the templates on the entire customer base. We use the term "champion / challenger" to reference this functionality.

We'll use an example to explain. Let's assume your prevailing overdue process template for residential customers starts with a "gentle reminder" letter followed 10 days later by a letter threatening collection agency referral if payment is not received. You may want to experiment with the impact of a change to this template. For example, you may want to change the "gentle reminder" to something more assertive and follow this up 5 days later with an even sterner warning. You can use the "champion / challenger" functionality to perform this experiment.

The following points describe how to implement "champion / challenger" functionality:

- Set up a "challenger" overdue process template for each template that you want to experiment with.

- Insert a new **Champion/Challenger** option on the Overdue Processing [Feature Configuration](#) for every champion template. Each option's value defines:
 - the "champion" overdue process template code
 - the "challenger" overdue process template code
 - the percentage of the time the system should use the "challenger" template
- Keep in mind that you can only experiment with one challenger template per champion template. For example, let's assume you have two prevailing overdue process templates - one for residential customers and another for commercial customers. You can experiment with different challenger templates for the residential and commercial templates. However, you cannot experiment with two different challenger templates for the residential champion template (i.e., a champion template can have 0 or 1 challenger template).

After setting up the above, your implementation's [Overdue Rule Plug-In](#) may include logic to use the challenger template X % of the time rather than the champion template. The sample plug-in provided in the base product, called [C1-CB-CR-RAT](#), includes this logic.

If you are using the Oracle Utilities Analytics product, you can configure analytic zones in innumerable ways to compare the efficacy of the champion versus the challenger. For example,

- You can set up a graph to show the average duration of each type of process.
- You can set up a graph to show the average dollars that were successfully collected.
- You can set up a dimensional scorecard to show how each template performed in different regions (or customer classes or ...).
- Etc (the list is limited by your imagination)

Overdue Process Information Is Overridable

"Overdue process info" is the concatenated string of information that summarizes an overdue process throughout the user interface. The base-package logic constructs this string by concatenating the following information:

- The description of its overdue process template
- Its status
- For active processes, the number of days since it was created. For inactive processes, the number of days since it was inactivated.
- For active processes, the unpaid amount of the objects being collected

If you'd prefer a different info string, you can develop a new algorithm and plug-it in on your [overdue process templates](#). This design allows some / all overdue process templates to have an override info string.

Original and Unpaid Amounts

There are two amounts associated with each overdue object linked to an overdue process: its Original Amount and its Unpaid Amount. These amounts are used throughout overdue processing.

You control how these amounts are calculated by defining the appropriate algorithm on your [overdue process templates](#). For example, you can plug in a base-package algorithm ([C1-CUAOA](#)) if you collect on overdue bills and the original amount is the amount of financial transactions linked to the bill, and the unpaid amount is the sum of financial transactions that are not linked to balanced match events.

Overdue Processes Are Highlighted Elsewhere

The topics in this section describe how other parts of the system highlight the existence of overdue and cut processes.

Alert Zone

If you plug-in the appropriate alert algorithm ([C1-OD-PROC](#)) on the Installation record, alert(s) will be shown for active overdue processes in the Alert Zone that appears in the Dashboard and on Control Central - Account Information.

Credit and Collections Zone

The Credit and Collections zone on [Control Central - Account Information](#) shows active overdue and cut processes.

Account Activity History Zone

The Account Activity History Zone on [Control Central - Account Information](#) shows pending and waiting events and inactive processes.

How Are Overdue Processes Cancelled?

A user may cancel an overdue process at their discretion, online / real-time using [Overdue Process - Main](#).

The system will automatically cancel an overdue process when the object(s) associated with the overdue process are sufficiently paid. Exactly when the system checks if an overdue process should be cancelled is dependent on your organization's billing and accounting rules. For example, if you practice [open-item accounting](#), you'd want to analyze an account's active overdue processes whenever a match event is added, changed or deleted (as match events are the only objects that impact if debt is considered paid in an open-item world). The base-package supports this specific example. If you need additional events to check if an overdue process should be canceled (e.g., the creation of a pay plan), a base-package change MAY be necessary. Please check with customer support if you have questions.

Two algorithms plugged-in on the [overdue process template](#) handle the cancellation:

- The Cancel Criteria algorithm is responsible for determining if an overdue process should be canceled. Algorithms of this type analyze the outstanding debt on the objects linked to the overdue process and indicate whether a process can be cancelled.
- The Cancel Logic algorithm is responsible for actually canceling the process. The logic involved in cancellation can be quite sophisticated as canceling an overdue process can result in the cancellation of its pending events and cut processes.

NOTE:

Why two algorithms? The reason two algorithms are involved in cancellation is because we want the cancellation logic to be encapsulated in one place so it can be called during both manual and automated cancellation.

Different logic for different templates. Because both the Cancel Criteria and Cancel Logic algorithms are plugged-in on the overdue process's template, you can have different cancellation criteria and logic for different templates.

Overdue Processes Are Created From Templates

As described above, you set up [overdue process templates](#) to define the types of events and when they are executed. When an overdue process is created, its events are created by copying the event types from an overdue process template. The remaining topics in this section provide background information to assist you in setting up your templates.

The Big Picture Of Overdue Events

This section describes the various types of overdue events and their lifecycle.

How Are Overdue Events Created?

Overdue events are created as follows:

- The [Overdue Monitor](#) invokes Overdue Monitor Rules to periodically check your accounts (refer to [Overdue Rules Are Embodied In Algorithms](#) for how this works). An Overdue Monitor Rule creates an overdue process when an account violates your overdue rules. The overdue process has one or more overdue event(s). The number and type of events is controlled by the overdue process template specified on the Overdue Monitor Rule.
- Users can create an overdue process manually on [Overdue Process - Main](#). To do this, they specify an overdue process template. The number and type of overdue events is defaulted from the template.
- An overdue event may be manually added to an existing overdue process by a user on [Overdue Process - Events](#).

NOTE:

Bottom line. Most overdue events are created by the system when it creates an overdue process for delinquent obligations. If you need to create an ad hoc overdue event, you can either create an overdue process whose template contains the desired event OR link the desired event to an existing process.

Overdue Events Can Do Many Things

An overdue event can perform a wide number of activities as the logic is embodied in an algorithm. The following points describe how this works:

- Every overdue event references an [overdue event type](#).
- The overdue event type, in turn, references an Event Activation algorithm.
- The Event Activation algorithm is invoked when the event is [triggered](#).

Overdue Event Information Is Overridable

"Overdue event info" is the concatenated string of information that summarizes an overdue event throughout the system. The base-package logic constructs this string by concatenating the following information:

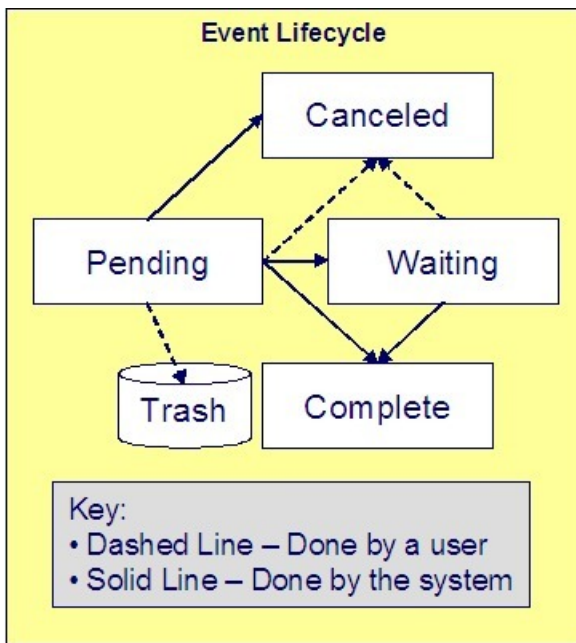
- The event type's description
- The event's status

- If it's pending:
 - If the event has a trigger date, the number of days until it's triggered plus the verbiage day(s) from today
 - Otherwise, the verbiage dependent on other events
- If it's waiting, the number of days, hours and minutes that it's been waiting
- If it's canceled, the cancel reason code's description
- If it's complete, the number of days, hours and minutes that it's been complete

If you'd prefer a different info string, you can develop a new algorithm and plug-it in on your event types. This design allows some / all event types to have an override info string.

Overdue Event Lifecycle

The following diagram shows the possible lifecycle of an overdue event:



Overdue events are initially created in the Pending state. An event can take myriad paths after it's created; it all depends on how you've configured the system. The following topics describe an event's lifecycle:

How and When Events Are Activated

An overdue event contains the date it should be activated; this is referred to as its trigger date. On this date, the Overdue / Cut Event Manager (a background process (C1-ODET)) invokes the Event Activation algorithm plugged-in on the event's event type. The Event Activation algorithm, in turn, will decide on the state in which to leave the overdue event (e.g., it may transition it to the Complete state or the Waiting state).

If a user can't wait for the Overdue / Cut Event Manager real-time, they can click a button on [Overdue Process - Main](#) to activate (and recursively trigger) events online / real-time.

You control how an event's trigger date is populated by configuring the [overdue process template](#). You are given two choices when you link an event type to an overdue process template:

- You can indicate the event should be assigned a trigger date when it is first created. You'd use this approach on the first event and events with no dependencies on earlier events. The following points describe how to configure the overdue process template to do this:
 - Indicate the event type is NOT dependent on other events, and
 - Define the number of days after the process's creation to use when calculating the trigger date.
- You can indicate the event should be assigned a trigger date only after earlier events are Complete. This technique should be used whenever you have an event that is only executed after other events are Complete. The following points describe how to configure the overdue process template to do this:
 - Indicate the event is dependent on other events, and
 - Define the number of days after the completion / cancellation of all dependent event(s) that the trigger date should be set to. The Overdue / Cut Event Manager sets the trigger date on such an event when it detects that all of its dependent events are complete / canceled.

NOTE:

Calendar vs Workdays. When an overdue event is created by the system, its trigger date is set in accordance with your date arithmetic preferences. Refer to [Calendar vs. Work Days](#) for more information.

Activating Events Should Add a Log Entry

As described [above](#), an overdue process has a log holding a history of meaningful events in the process's life. Most Event Activation algorithms will add an entry to the process's log.

These log entries are more than just an audit trail as they also reference the objects that are created during activation. For example, if an activation algorithm creates a customer contact, the ID of the customer contact will be referenced on the log (and end-users will be able to drill down on the log entry to see the customer contact).

Holding Events

You can prevent a pending event with a trigger date on / before the current date from activating by plugging-in a Hold Event Activation plug-in on the overdue process template. This might prove useful, for example, if you want to suspend an overdue process while a [case](#) used to investigate the assertion of a customer is outstanding. Then, when the case closes, the overdue process can start up where it left off.

This technique would prove useful, for example, if your users can grant ad hoc suspend periods (e.g., if a customer wants a few extra days to pay before a cutoff). To do this, create a case type that has two states: Open and Close. Make sure to set up the Open state to have an automatic transition algorithm to close the case after X days. Then, all a user has to do is create a case of this type when they want to provide a suspend period. When the suspend period is over and payment isn't received, the case will close and the overdue process will start up where it left off.

Some Events Wait For Something Before They Activate

Consider this scenario - you want an overdue event to create a To Do entry so a user can authorize the next phase of an overdue process. When this event activates, the event's activation algorithm will create a To Do entry, but it will NOT transition the event to complete. Rather, the overdue event will exist in the waiting state. While in the waiting state, the Overdue / Cut Event Manager will monitor the state of the To Do entry. When the To Do entry completes, the original overdue event can transition to the complete state and then latter dependent events can be triggered. The following points describe how to configure the system to support this type of event:

- The event type's Event Activation algorithm should behave as follows:
 - It creates the object on which the overdue event waits.
 - It must link this object to the overdue process by creating a log entry where the prime-key of the related object is referenced (in a foreign-key characteristic). This log entry should also reference the event.
 - It should leave the overdue event in the waiting state.
- The event type must have a Monitor Waiting Event algorithm. This algorithm is invoked each time the [Overdue / Cut Event Manager](#) executes. If the related object has transitioned to a "final" state, the originating overdue event is transitioned to the complete state (and then latter dependent events are triggered).

NOTE:

Bottom line. Two algorithms must be set up on an overdue event type to implement waiting functionality: an Event Activation algorithm that creates the monitored object and a Monitor Waiting Event algorithm to check on the state of the monitored object. The Overdue / Cut Event Manager has the dual responsibility of activating the event and monitoring its related object for completion (and then triggering the dependent events when it completes).

While the above example illustrated how an overdue event could create and then monitor a To Do entry, you can use this functionality to create and monitor any object that has an initial and final state. If the base package does not contain the algorithms you need, simply develop new ones using the base-package algorithms as examples.

How Are Events Canceled

A pending event will be cancelled automatically by the system when the overdue process is canceled. Refer to [How Are Overdue Processes Cancelled](#) for more information.

A user may cancel a pending or waiting event at their discretion.

Regardless of what triggers the cancellation, the Cancel Logic algorithm plugged in on the overdue event type handles the cancellation. This allows you to introduce additional cancellation logic should the need arise. Please note that the base package cancel algorithms insert a [log entry](#) when a user manually cancels an event.

The Big Picture Of Cut Processes

While not required, many overdue processes will contain one or more events that will cut (i.e., stop) service. When such an event activates, it creates a "cut process". A cut process is very similar to an overdue process; the major difference is that a cut process is linked to a specific service agreement (the one being cut), whereas an overdue process is linked to the object(s) in arrears.

The topics in this section provide background information about cut processes.

How Are Cut Processes Created?

The activation of an overdue event can create one or more cut processes.

In addition, a user can manually create a cut process at their discretion using [Cut Process - Main](#).

Overdue Events Wait For The Cut Process To Conclude

Overdue events that create cut processes are perfect examples of events that [wait](#) for the object they create to complete before they, in turn, complete. After the cut process concludes, the originating overdue event will complete thus triggering its dependent events.

Cut Processes Exist Under An Overdue Process

It's important to note that a cut process exists in respect of an overdue process. In other words, you can't create a cut process without referencing a parent overdue process. There are two reasons why:

- A cut process's overdue process defines the objects in arrears. It is these objects that are monitored to determine if the cut process (and overdue process) can be cancelled.
- The [log](#) associated with a cut process's overdue process contains the history of the cut process(es) and their events; there is no log specific to a cut process.

The Components Of A Cut Process

Cut processes are simpler than overdue processes as they simply contain one or more "cut events". These events are the service agreement-specific actions designed to encourage the customer to pay. For example, you might set up cut events that:

- Create [field activities to cut service](#)
- Send letters (via the creation of a customer contact)
- Create To Do entries
- Impact the account's credit rating
- Create a case (e.g., to manage a customer with life support issues)
- ... (the list is only limited by your imagination as algorithms are used to perform the event's actions)

You define the number and type of events by configuring [cut process templates](#). When the system creates a cut process, it copies the events defined on the related template.

It's important to note that all cut events are created when the cut process is created. The Overdue / Cut Event Manager is responsible for activating the cut events at the appropriate time. Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do entry to a user, create a field activity, etc.)

Cut processes do not have their own log as their history is maintained on the parent overdue process's [log](#).

Cut Process Information Is Overridable

"Cut process info" is the concatenated string of information that summarizes a cut process throughout the system. The base-package logic constructs this string by concatenating the following information:

- Its cut process template's description
- Its status
- For active processes, the number of days since it was created. For inactive processes, the number of days since it was inactivated.
- For active processes, the unpaid amount of the objects being collected on the cut process's overdue process

If you'd prefer a different info string, you can develop a new algorithm and plug-it in on your cut process templates. This design allows some / all cut process templates to have an override info string.

How Are Cut Processes Cancelled?

A user may cancel a cut process at their discretion, online / real-time using [Cut Process - Main](#).

The system will automatically cancel a cut process when the overdue process is cancelled.

Regardless of what triggers the cancellation, the Cancel Cut Process algorithm plugged in on the cut process's [template](#) handles the cancellation.

If a cut process is canceled and the cut process's events created field activities, the cancel logic embodied in the algorithm can be quite sophisticated. Refer to [Field Activities To Cut And Reconnect Service](#) for an example.

Cut Processes and Events Are Created From Templates

When a cut process is created, its cut events are created by copying a cut process template's event types. Cut events follow the same lifecycle and possess the same behavior as [overdue events](#).

Cut Events Are Like Overdue Events

Cut events almost identical to [overdue events](#). The major difference is that cut events exists under a cut process and therefore are oriented towards the cut process's service agreement. The following topics describe other differences between cut events and overdue events.

Field Activities To Cut and Reconnect Service

A cut process that's created for a service agreement whose service can be severed will typically contain a cut event that creates field activity(s) to cut service. The base-package cut event activation algorithm will create field activities for the service points linked to the service agreement being cut. The type of activity is defined on the [field activity type profile](#) defined on each service point's SP type. Please see [C1-CE-CR-FA](#) for the exact details of the base-package algorithm.

If the cut process is canceled after field activity(s) are created and before the cut process completes, different activities can transpire. The specifics of what happens are controlled by a Cancel Logic algorithm plugged in on the [cut process template](#). The following points summarize how a base-package algorithm ([C1-CP-DWFA](#)) works:

- It assembles all Pending, Held and Complete cut for nonpayment field activities (FA) created by the cut process (these are defined in the overdue process's [log](#)).
- Each FA is processed as follows:
 - If the FA is Pending or Held and it is not linked to a field order (i.e., it hasn't been dispatched):
 - The FA is cancelled
 - An entry is added to the overdue process log to indicate the cancellation
 - If the FA is Pending or Held and it is linked to a field order (i.e., it has been dispatched):
 - A To Do entry is created (the To Do Type is defined on the algorithm)
 - If the FA is Complete and the Cut Process's SA is pending start or active (i.e., it hasn't been expired yet):
 - Reconnect field activity(s) are created for every service point that had a cut for non-payment field activity. The type of activity is defined on the [field activity type profile](#) defined on each service point's SP type.

A Cut Event Expires Service Agreements (and May Trigger Final Billing)

The last cut event is typically set up with an activation algorithm that expires the cut process's service agreement. If earlier cut events created "cut for non-payment" field activities, these field activities will be used as the basis for stopping service. Refer to [Finalizing Pending Stops](#) for how the system uses the meter reads on these field activities as the "stop reads" on the service agreement. Note, you can see the field activities that are used to "cut" and "stop" service by viewing the Field Activities grid on [Service Agreement - Service Point](#).

NOTE:

Final billing. When the last service agreement linked to an account is expired, the system will schedule the account for billing (outside of its normal bill cycle schedule).

Write Offs Are Implemented Using Overdue Events

The system has been designed to allow overdue events on the original overdue process to write-off the objects being collected. Refer to [Writing Off Bills](#) for a case-study explaining how to do this.

Calendar vs. Work Days

When you set up your overdue and cut process templates, you supply information that controls how each event's trigger date is calculated. You have two options:

- You can say that an event's trigger date can only be populated after earlier, dependent events are complete. For example, the 2nd event (cut service) is triggered 2 days after the 1st event is complete (48 hour warning letter).
- You can say that an event's trigger date is populated when the process is first created. You simply define the number of days after the start of the process when each such event should be triggered. For example, the 2nd event (send cutoff warning) can be triggered 7 days after the start of the process.

In addition to the above, an option defined on the [Feature Configuration for Overdue Processing](#) plays a part in the calculation of an event's trigger date:

- If you set the option to use calendar days, the trigger date of events will be set to the first workday on / after the calculated date. For example, if you indicate that the 2nd event is triggered 7 days after the 1st event, the system will add 7 days to the 1st event's completion date. It then checks if this is a workday (and not a holiday); if so, this is the trigger date of the event; if not, it assigns the trigger date to the next workday.
- If you set the option to use workdays, the trigger date will be calculated by counting workdays. For example, if you indicate that the 2nd event is triggered 7 days after the 1st event, the system will count 7 workdays and set the trigger date accordingly.

NOTE:

Account's division controls the work calendar. The system uses the above information in conjunction with the [work calendar](#) associated with the account's division to determine workdays.

Bill-Oriented Collection - Advanced Topics

The topics in this section provide information on features designed to facilitate the collection of overdue bills.

Miscellaneous Bill-Oriented Collection Topics

The topics in this section provide background information about a variety of bill-oriented collection topics.

Highlights Of The Sample Overdue Monitor Rule Algorithm

A base-package Overdue Monitor Rule algorithm exists to support many different types of overdue rules (see [C1-CB-CR-RAT](#)). Keep in mind that multiple such algorithms can be plugged in on [Collection Class Overdue Rules](#). When the [Overdue Monitor](#) analyzes an account, it calls these algorithms until there are no more to invoke (or until an algorithm tells it that there is nothing more to check). The following points describe its various options:

- **Different rules based on credit ratings.** You can set up algorithms to only be applied to accounts with a credit rating \leq a given value. We anticipate such algorithms will be used in conjunction with other such algorithms to apply different rules based on the customer's credit rating.
- **Different rules based on an account or service agreement characteristic.** You can set up algorithms to only be applied if an account or one of its service agreements has a given characteristic type and value effective within the last X days. For example, you could set up an algorithm that would process an account if it had a broken payment arrangement service agreement within the last 365 days (broken payment arrangement service agreements are marked with a given char type and value). You could set up a different algorithm that treated strategic customers more leniently (given that the definition of "strategic" could be held in an account or service agreement characteristic).
- **Skipping a bill if it has a future postpone date.** You can set up algorithms to ignore an unpaid bill if it has a "postpone date" that's in the future. This date would be defined on the bill using an ad hoc characteristic. The characteristic type is defined as a parameter on the algorithm.
- **Special process for bills with disputed debt.** You can set up an algorithm to create a given type of overdue process if the bill has disputed debt. By "disputed debt", we mean a bill with a financial transaction that's linked to an unbalanced match event that's marked as Disputed.
- **Special process for bills with credit balances.** You can set up an algorithm to create a special overdue process if the bill's total charges are negative (i.e., a credit bill). Such bills should be very unusual as we strongly recommend that credit bills should have their credit balance applied to an overpayment service agreement during bill completion.
- **Different rules when all service agreements are inactive.** You can set up an algorithm that should only be applied if all service agreements are inactive. We anticipate this can be used to differ the type of overdue process as if an account does not have active service agreements; there is nothing to stop to encourage them to pay.
- **Different rules based on age and amount of a bill.** You can set up an algorithm to create different types of overdue processes based on the age and amount of the bills.

NOTE:

You are not limited by the base-package's rules. If your implementation requires options that are not supported by the base-package algorithm, simply develop your own.

Holding A Process's Events

Refer to [Holding Events](#) for a description of how to prevent the activation of an overdue / cut processes events while a given condition is true. The condition is defined in an algorithm so you can set up the system to prevent event activation for practically any condition. For example, you might want to hold event activation while the bill's account has an active case (e.g., a user might set up a case to conduct an ad hoc investigation of billing discrepancies). When the case closes, you'd want the process to start up where it left off.

Bill Info Shows Unpaid and Write-off Amounts

The base-package Bill Information algorithm (plugged in on the [Installation](#) record) is responsible for constructing a bill's summary information that appears throughout the system. This algorithm can be configured to show the unpaid and write-off amounts of each bill.

Writing Off Bills

The topics in this section provide background information on how to write off bills.

After Service Is Cut, Write-Off Oriented Events Can Commence

After the last service agreement has been stopped, overdue events to manage the write-off of the overdue process's bill(s) should be triggered. Please note that a separate overdue process is NOT created to manage write-offs. Rather, events associated with the original overdue process will handle the write-off activities.

Collection Agency Referrals

Before debt is written off, many implementations refer the unpaid bills to a collection agency. The following points describe how to implement this.

- Set up an overdue event type with an activation algorithm that refers an overdue process's bill(s) to a collection agency.
- When such an event is activated, the system creates a [Collection Referral](#) record for a collection agency. The specific agency to which the debt is referred is controlled by the event type's activation algorithm. The sample algorithm simply refers debt to the collection agency with the least amount of referred debt. If you prefer different logic, you must develop your own algorithm.
- A collection agency referral history record is linked to an account. It contains the amount of debt referred to the collection agency. It is the creation of this record that triggers the interface of information to the collection agency. The method used to interface the information to the agency is defined on the collection agency's record. Refer to [Setting Up Collection Agencies](#) for more information.
- If the collection agency is successful in obtaining the funds, a payment will be added. If the payment satisfies the cancel criteria defined on the overdue process template's cancellation plug-in, the overdue process will cancel. When an overdue process is cancelled, the cancel criteria on the overdue process's template are executed. We strongly recommend plugging in an algorithm that will cancel collection agency referrals when an overdue process is cancelled.
- If the collection agency is not successful in obtaining your funds after a given amount of time, you probably want to cancel the referral and write-off the debt. The cancellation of the referral will happen automatically if you set up your overdue process template to have an event that creates a collection agency cancellation X days after the referral. You can cancel a referral manually by simply creating a new collection agency referral history record (with a type of cancel).
- Collection agencies are notified of the cancellation of a referral by the creation of a new collection agency referral history record (with a type of cancel). This record will be interfaced to the agency in the same manner used to interface a new referral (see above).

NOTE:

Log entry. The base-package overdue event activation algorithms that make and cancel collection agency referrals insert rows in the overdue process's [log](#) to audit these events.

Writing Off Bills Will Create Write-Off Adjustments And Possibly A Match Event

Most overdue process templates will be configured to contain an event that writes-off the unpaid balance of bills.

NOTE:

Processing is redirected to another algorithm. The base-package overdue event activation algorithm that writes off bills simply redirects the call to the Write-Off Bill algorithm plugged in on the account's [Collection Class Overdue Rules](#). The reason for this redirection is because users can manually write-off a bill (using [Bill - Main](#)) and when they do this, we want to invoke the same logic as when an overdue process writes-off a bill.

The base-package Collection Class Overdue Rules - Write-Off Bill algorithm determines the amount that should be written off for each distribution code on each unpaid financial transaction. It then creates a separate adjustment for each service agreement where the lines on the adjustment contain the amount to be written off for each distribution code. If the unpaid financial transactions are not linked to a match event, the write-off adjustments plus the unpaid financial transactions will be linked to a new match event. If the unpaid financial transactions were linked to an unbalanced match event, the write-off adjustments will be added to the existing match event (thus making it balanced).

If partial payments were made against a bill, the amount written off will be prorated in light of the partial payments. For example, if 20% of a bill had been paid, 80% of each distribution code will be written off.

For example, assume the original bill's FT looked as follows (note, if the bill has multiple service agreements this will need to be done for each SA's FT's):

- Debit A/R \$110.00
- Credit Flat Charge Revenue (\$50)
- Credit Usage Revenue (\$50)
- Credit City Tax Payable (\$5)
- Credit State Tax Payable (\$5)

Assume the customer had made a partial payment of \$11 and it was matched to this FT. At write-off time, the system will create an adjustment whose adjustment lines will cause each distribution code to be written off by 90% (100% - \$11 / \$110). For example:

- Debit Flat Charge Revenue (or some W/O Expense) \$45
- Debit Usage Revenue \$45
- Debit City Tax Payable \$4.50
- Debit State Tax Payable \$4.50
- Credit A/R \$99

This adjustment will then be linked to the original match event on which the payment was linked (thus making it Balanced).

NOTE:

Log entry. The base-package overdue event activation algorithm that writes off bills inserts a row into the overdue process's [log](#) for each bill written off.

Small Amount Write-Downs

Many organizations will write-down a bill whose value is small early in an overdue process. The base-package overdue event activation algorithm has parameters to support this requirement (this algorithm allows you to write-off an overdue process's bill(s) if their value is less than a threshold amount).

If your organization writes-down small amounts differently than large amount, simply set up an overdue event type to reference such an activation algorithm and position it in the appropriate place in the overdue process template.

Bill Info Shows Written Off Amounts

The base-package Bill Information algorithm (plugged in on the [Installation](#) record) is responsible for constructing a bill's summary information that appears throughout the system. This algorithm can be configured to show the amount written-off for each bill.

Online Write Off Of Bills Is Performed On Bill-Main

If a bill's account is associated with a [Collection Class Overdue Rule](#) that has a Write Off Bill algorithm, a button appears on [Bill - Main](#) if the bill hasn't been written off. When clicked, the Write Off Bill algorithm is invoked to write-off the bill.

Online Reversal Of Written Off Bills Is Performed On Bill-Main

If a bill's account is associated with a [Collection Class Overdue Rule](#) that has a Write Off Bill algorithm, a button appears on [Bill - Main](#) if the bill has been written off. When clicked, the Write Off Bill algorithm is invoked (in reversal mode). The algorithm cancels the write-off adjustments (thus making the bill payable again). It also schedules the account for review by the [Overdue Monitor](#) the next time it runs.

Write Offs And Overdue Process Cancellation

It should be stressed that writing-off a bill may cause an overdue process to be canceled because the bill's FT will be linked to a balanced match event (note, the specific [cancel criteria](#) are in a plug-in so this is not a hard rule).

The following points highlight interesting aspects of bill write-off and overdue process cancellation:

- If a user writes off a bill [real time](#) and the bill has an Active overdue process, the process's cancel criteria will be invoked. This typically results in the cancellation of the overdue process.
- If an overdue event writes off a bill, the state of the process depends on your cancel criteria and where the overdue event is positioned in the overdue process. For example,
 - Imagine an overdue process that has an overdue event that writes offs small amounts of debt early in the process. If such a process is applied against bill with a small amount of debt, the process will be canceled (when the event activates).
 - Contrast this to an overdue process where the last event writes off the bill. Because there are no other events to activate, the process will complete (i.e., it will not be canceled).

An Alert Can Show Written Off Bills

A base-package Control Central Alert algorithm ([C1-WO-BILL](#)) can be set up to highlight if the account in context has written off bills. This algorithm is plugged in on the [Installation](#) record.

Bill-Oriented Payment Arrangements

A payment arrangement is an agreement with a customer to payoff severely overdue debt in **billed** installments. Bills sent to customers with payment arrangements contain charges for both their current services and their payment arrangement installment amount.

NOTE:

Nomenclature. Some organizations refer to payment arrangements as "current bill plus" agreements because the customer's bills contain charges for both their current debt plus their installment amount. After the customer has paid off their overdue debt, the payment arrangement closes and the customer's bills only contain charges for their current debt.

The topics in this section describe how to set up a payment arrangement and how the system monitors the ongoing arrangements.

Creating Payment Arrangements

When you create a payment arrangement, you are actually creating a service agreement. This service agreement is just like other service agreements in that:

- It holds debt.
- It is periodically billed (thus creating unmatched bill segment financial transactions).
- When a payment is received, the payment segment financial transactions are matched to the bill segment financial transactions.

Debt is transferred to a payment arrangement service agreement (PA SA) from the customer's delinquent bills at the inception of the payment arrangement.

When you transfer debt from the overdue bills to a PA SA, transfer adjustments are created to transfer debt from the delinquent SAs to the PA SA. Match events are created to link the "transfer from" adjustments to the original unpaid financial transactions (FT). When all FT's on a bill are linked to balanced match events, the bill is no longer considered overdue and any active overdue process (and its related cut processes) will be cancelled. Refer to [How Are Overdue Processes Canceled](#) for more information.

NOTE:

Use the Payment Arrangement Transaction. Use the [Payment Arrangement - Bill Oriented](#) to set up bill-oriented payment arrangements. This transaction creates a PA SA, transfers overdue bills to it, and sets up the installment amount. This transaction is also used if you need to break or cancel the payment arrangement.

Installment, Payoff and Current Amounts

WARNING:

If you do not understand the difference between payoff balance and current balance, refer to [Current Amount versus Payoff Amount](#).

When you set up a payment arrangement service agreement (PA SA), you transfer delinquent debt to the PA SA using transfer adjustments. After moneys are transferred, the system sets the PA SA's current balance to zero. At this point, there will be no overdue bills. If the customer neglects to pay bills containing charges associated with the payment arrangement, an overdue process will ensue.

PA SAs start their life with a non-zero payoff balance (i.e., they have debt when first started). This debt is transferred from the bills whose outstanding debt necessitated the creation of the PA SA.

The installment amount that the customer is billed is determined by the number of installments used to payoff the debt. For example, if the customer owes \$500 and they want to pay this off in 10 installments, you'd set up the installment amount to be \$50. The installment amount is saved on the PA SA's recurring charge amount. If the customer again falls into arrears on their bills, you can transfer additional bills to the PA SA. You can also change the installment amount as needed.

A PA SA's payoff balance typically differs from its current balance. The payoff balance is the amount of debt remaining to be paid off under the terms of the payment arrangement. The current balance is the installment amount that has been billed but not paid. For example, a customer who is paying off \$500 with 10 installments of \$50 would have an initial payoff balance of \$500 and a current balance of \$0. After the first bill, the PA SA would still have a payoff balance of \$500, but its current balance would be \$50. When the customer pays, the PA SA's payoff balance would fall to \$450 and its current balance would return to \$0.

The following table contains a financial example of a customer who sets up a payment arrangement to payoff \$1,000 of debt in \$10 installments.

Event	Normal SA's GL Accounting	PA SA's GL Accounting	Normal SA's Current Balance	Normal SA's Payoff Balance	PA SA's Current Balance	PA SA's Payoff Balance
Prior to creation of payment arrangement	N/A	N/A	1000	1000	N/A	N/A
Transfer debt from normal SA(s) to PA SA	Xfer 1000 A/R <1000>	PA A/R 1000 Xfer <1000>	0	0	1000	1000
Set current balance to zero on PA SA	N/A	N/A	0	0	0	1000
Customer is billed (\$50 for new debt and \$10 of payment arrangement debt)	A/R 50 Revenue <50>	N/A	50	50	10	1000
Customer pays \$60	Cash 50 A/R <50>	Cash 10 PA A/R <10>	0	0	0	990

When the customer pays off the payment arrangement debt, the system automatically closes the PA SA after it final bills (assuming the PA SA's SA type references a bill segment type that has a bill segment creation algorithm of Recurring Charge With Auto Stop).

Breaking A Bill-Oriented Payment Arrangement

If a customer neglects to pay a bill, an overdue process is created. If the bill contains charges from a payment arrangement, you'll want to "break" the payment arrangement. By "break", we mean cancel the arrangement and reinstate the debt under the originating bills. To do this, you must configure the [cut process template](#) used to cut the PA SA to contain a cut event that breaks the payment arrangement.

NOTE:

Processing is redirected to another algorithm. The base-package cut event activation algorithm that breaks a payment arrangement simply redirects the call to the Bill-Based Payment Arrangement algorithm plugged in on the account's [Collection Class Overdue Rules](#). The reason for this redirection is that users can manually break a payment arrangement (using [Payment Arrangement - Bill Oriented](#)). When they do this, we want to invoke the same logic as when a cut event activation algorithm breaks a payment arrangement.

The base-package Collection Class Overdue Rule - Bill-Based Payment Arrangement algorithm does the following:

- Creates "cancel adjustments" for all unpaid financial transactions associated with the payment arrangement (e.g., billed, but not paid installments).
 - Cancels ALL adjustments that were used to transfer the debt to the payment arrangement. When these are cancelled, the original bills will become unpaid (and the debt will be rather old by this point).
 - Syncs up current balance with payoff balance on the PA SA.
 - Makes the PA SA pending stop and expires it (SA activation will stop the SA when it next runs).
 - If there is a credit left on the PA SA (because payments were made against the arrangement), the credit will be distributed amongst the bills that contributed debt to it (the oldest bills are paid off first). This relief will occur via transfer adjustments from the PA SA to the original SAs.
 - Note, if there is a debit left (e.g., because LPC were issued or some other type of adjustment was created by an operator), an error is issued as we don't handle new debit on a payment arrangement.
 - Inserts a characteristic on the PA SA to indicate that it has been broken (you might want to use this to cause more strict overdue rules to be applied to accounts with broken payment arrangements).
 - Marks the account so it will be reviewed by the [Overdue Monitor](#) the next time it runs.
-

NOTE:

The PA SA must final bill before it closes. It's important to note that the PA SA will only close after the PA SA is final billed. This is OK as it won't have any money left on it.

When the Overdue Monitor next runs, it will analyze the account's reinstated bills. We recommend setting up a Collection Class Overdue Rule - Overdue Monitor Rule algorithm that has more stringent rules if the account has a broken payment arrangement within the last X days. The base-package algorithm ([CI-CB-CR-RAT](#)) supports this type of processing.

NOTE:

Log entry. The base-package overdue event activation algorithm that breaks a payment arrangement SA inserts a row into the overdue process's [log](#).

Online Breaking

A user can click a button on the [Payment Arrangement - Bill Oriented](#) page to break a payment arrangement real time. When clicked, the Collection Class Overdue Rule - Bill-Based Payment Arrangement algorithm is called to break the payment arrangement. This is the same algorithm called when a cut event break a payment arrangement.

Online Canceling

A user can click a button on the [Payment Arrangement - Bill Oriented](#) page to cancel a payment arrangement real time. Cancellation should be used when you want to "logically delete" a PA SA because it shouldn't have been created.

The logic described above for breaking a payment arrangement is executed when a user cancels a payment arrangement; the only difference is that the PA SA is marked with a different characteristic type / value than when it is broken. The "broken" characteristic type / value can be used to apply stricter rules to the account when it's next reviewed by the [Overdue Monitor](#).

Creating Overdue and Cut Procedures

Your overdue procedures define how your organization collects overdue debt. Your cut procedures define how your organization cuts (i.e., stops) service agreements when collection attempts fail. In this section, we describe how to set up the data that controls these procedures.

WARNING:

There are numerous ways to design your overdue and cut procedures. Some designs will result in easy long-term maintenance; others will result in maintenance headaches. In this section, we provide information to help you understand the ramifications of the various options. Before you set up your overdue and cut procedures, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

Designing Your Overdue Procedures

The design of your overdue procedures is an iterative process. Over time, you will develop skills that will allow you to skip some steps. However, when you're starting out, we recommend you use the following matrix as your guide. When the matrix is complete, you're ready to set up the overdue processing control tables.

Account's Division	Account's Collection Class:	Account's Collection Class:

The topics discussed below will gradually complete this matrix using a simple case study.

FASTPATH:

For more information about how the information in this matrix is used to monitor your customers' debt, refer to [How Does The Overdue Monitor Work](#).

Your Divisions Are Frequently Preordained

An account's division defines the jurisdiction whose rules govern the account. For example, an account's division controls how its payments are distributed, if / how late payment charges are levied, etc. Divisions have typically been designed in advance of designing your overdue rules.

In our example, we assume you have two divisions: North and South.

Account's Division	Account's Collection Class:	Account's Collection Class:
North		

Designing Your Collection Classes

Multiple collection classes are needed when your organization has different overdue rules and / or procedures based on the type of customer. If all customers are treated the same way, you'll have a single collection class (call it Generic). However, if you're like many organizations, you will have multiple collection classes.

For example, for commercial/industrial customers, you probably don't worry until they owe you more than, say, \$100 after 20 days. For residential customers, you probably don't worry until they owe you more than, say, \$5 after 20 days. In this situation, you will have at least two collection classes: one for commercial/industrial customers, the other for residential customers.

In our example, we assume you have two collection classes: Residential and Commercial/Industrial.

Account's Division	Account's Collection Class:	Account's Collection Class:
	Residential	Commercial/Industrial
North		
South		

NOTE:

There are two very different ways to monitor your accounts for overdue debt. This chapter describes the method referred to as Overdue Processing. Refer to [Collection / Severance / Write Off](#) for a description of the other method. We anticipate that most organizations will only use a single method. If your organization opts to use both methods, you will need to set up the corresponding collection classes.

Designing Overdue Monitoring Rules

At this point, we have the rows and columns defined in our matrix. Now it's time to work on the individual cells.

NOTE:

Single currency. We've assumed that your implementation works in a single currency. If this is not true, you will need to add a 3rd dimension that will have a value for each currency code.

Each cell will contain the rules that the [Overdue Monitor](#) uses to determine if an account has overdue debt. These rules will eventually be configured using one or more algorithms on [Collection Class Overdue Rules](#).

Account's Division	Account's Collection Class:	Account's Collection Class:
	Residential	Commercial/Industrial
North		
South		

NOTE:

If the Overdue Monitor encounters an account whose collection class and division does not have overdue rules set up, it will issue an error.

Determining the rules in each cell can be straightforward or complicated; it depends on how your organization works. Our case study assumes the following:

- For residential debt (regardless of division) we have the following rules:
 - Highest priority. If a bill exists with unpaid FT's > \$0 that is older than 50 days, create the "accelerated overdue process" for residential customers. We'll talk more about this process later.
 - Medium priority. If the account's has a broken payment arrangement within the last 60 days with an unpaid bill > \$0 that is older than 20 days, create the "broken payment arrangement overdue process".
 - Lower priority. If a bill exists with unpaid FT's > \$25 that is older than 25 days, create the "courtesy reminder overdue process" for residential customers. We'll talk more about this overdue process later.
- For commercial-industrial debt (regardless of division) we have the following rules:
 - Highest priority. If a bill exists with unpaid FT's > \$0 that is older than 45 days, create the "commercial 45 days late overdue process". We'll talk more about this process later.
 - Medium priority. If the account's credit rating is < 550 and has an unpaid bill > \$0 that is older than 20 days, create the "risky commercial customer overdue process".
 - Lower priority. If a bill exists with unpaid FT's > \$100 that is older than 30 days, create the "commercial 30 days late overdue process". We'll talk more about this overdue process later.

Given the above, our matrix will look as follows:

Account's Division	Account's Collection Class:	
	Residential	Commercial/Industrial
North	<p>Highest Priority: If a bill exists with unpaid FT's > \$0 that is older than 50 days, create the accelerated overdue process for residential customers.</p> <p>Medium Priority: If the account has a broken payment arrangement within the last 60 days with an unpaid bill > \$0 that is older than 20 days, create the broken payment arrangement overdue process.</p> <p>Lowest Priority: If a bill exists with unpaid FT's > \$25 that is older than 25 days, create the courtesy reminder overdue process for residential customers.</p>	<p>Highest Priority: If a bill exists with unpaid FT's > \$0 that is older than 45 days, create the commercial 45 days late overdue process.</p> <p>Medium priority. If the account's credit rating is < 550 and has an unpaid bill > \$0 that is older than 20 days, create the risky commercial customer overdue process.</p> <p>Lowest Priority: If a bill exists with unpaid FT's > \$100 that is older than 30 days, create the commercial 30 days late overdue process.</p>
South	Same as above	Same as above

NOTE:

The rules are limited by your imagination (and business requirements). While the base-package Overdue Monitor Rule algorithm ([C1-CB-CR-RAT](#)) supports the above scenarios, we'd like to stress these are just examples. Your implementation can operate using very different rules by either configuring the base-package algorithm (it has many parameters that you can use to tailor your rules) OR by introducing a new algorithm. Refer to [Highlights Of The Sample Overdue Monitor Rule Algorithm](#) for the options delivered with the base-package.

Designing Overdue Process Templates and Event Types

The following table shows a sample overdue process template for one of the rules in the Residential / North cell in the previous section's matrix.

Overdue Process Template	Overdue Event Type	When Triggered
Accelerated overdue process for residential customers	Old debt letter	At inception of process
	Cut active service agreements	10 days after inception
	Reduce customer's credit rating	10 days after inception (i.e., at the same time the cut process is created)
	Write down small debt	0 days after completion of the cut process(es)
	Refer debt to collection agent	0 days after attempting the small write down (this means that either the small write-down or the agency referral will take place as if the write-down is successful, the bill's FTs will be matched to balanced match events and the overdue process will stop)
	Cancel collection agent referral	45 days after referral
	Write-off debt	0 days after collection agent cancellation

You should create a similar table for each of the distinct overdue process templates in your matrix.

At this point, you've designed the distinct overdue process templates. Next, you'll need to design the algorithms that control their overdue processes:

- A template's Calculate Unpaid and Original Amount algorithm calculates the original and unpaid amounts of the objects being collected by a process. These values are used throughout the overdue processing module.
- A template's Cancel Criteria algorithm is executed to determine if a process should be cancelled. Refer to [How Are Overdue Processes Cancelled](#) for the details.
- A template's Cancel Logic algorithm is executed to cancel a process. Refer to [How Are Overdue Processes Cancelled](#) for the details. Please note that the logic embodied in this type of algorithm can be sophisticated because it is responsible for stopping an ongoing process's activities (e.g., this could involve cancelling field activities or cases). Cancellation algorithms are also responsible for inserting [log](#) entry(s).
- A template's Hold Event Activation algorithm is invoked to determine if the [Overdue / Cut Event Manager](#) should [suspend the activation](#) of the process's events.
- A template's Information algorithm is invoked to construct the [override "info string"](#).

Next, extract each unique event type from the above table:

Overdue Event Type	Action
Old debt letter	Create a customer contact
Cut active service agreement(s)	Start a cut process for every active SA with an unpaid FT on the bill
Reduce customer's credit rating	Insert an account credit rating history record
Write down small debt	Create write-down adjustments if unpaid debt is less than \$x
Refer debt to collection agent	Create a collection agency referral
Cancel collection agent referral	Cancel the collection agency referral
Write off unpaid debt	Create adjustments to write-off unpaid debt

At this point, you know the distinct event types. Next, you'll need to design the algorithms that control the lifecycle of each event type:

- The event type's Event Activation algorithm(s) are executed by the [Overdue / Cut Event Manager](#) on its trigger date. The following points describe the logic embodied in such an algorithm:
 - The activity that happens on the trigger date (e.g., creation of a customer contact, To Do, etc.). Refer to [Overdue Events Can Do Many Things](#) for the details.
 - Whether the event is transitioned into the Waiting or Complete state when it's triggered. Refer to [Some Events Can Wait](#) for the details.
 - How the log entry(s) associated with event activation will be constructed. The base-package algorithms allow you to control the verbiage in the log entry by defining the desired message number on the algorithm. This means that you may have to set up new messages. Refer to [Activating Events Should Add A Log Entry](#) for the details.
- The event type's Cancel Logic algorithm(s) are invoked when [an event is cancelled](#). The following points describe the logic embodied in such an algorithm:
 - If the event is allowed to be canceled. This logic may be necessary if some conditions prevent events of this type from canceling. For example, you may want to prevent an event from canceling when there are later dependent events that aren't canceled.
 - Any ancillary actions that take place during cancellation.
 - How the [log entry\(s\)](#) associated with event cancellation will be constructed.
- The event type's Monitor Waiting Event algorithm(s) are invoked to [monitor a waiting event](#). These algorithms are responsible for transitioning a Waiting event to Complete if the object on which it's waiting is complete.
- The event type's Event Information algorithm is invoked to construct the [override "info string"](#).

Once you've designed each event type's algorithms, you're ready to design your cut processes.

Designing Cut Process Templates and Event Types

While not required, many overdue processes will contain an overdue event that cuts (i.e., stops) service (in the hopes that stopping service will inspire the customer to pay). When such an event activates, it creates one or more cut process(es).

The system allows you to control if and how service is cut by setting up Cut Process Rule algorithms on your SA types. This allows different rules for different types of service agreements. In addition, because these rules are embodied in algorithms, a given SA type can have conditional logic that controls the type of cut process created. For example, you may have a different cut process if the customer has life support equipment, or if it's winter, or ...

NOTE:

If a service agreements of a given type should never be cut. If you have certain SA types that should never be cut, do NOT define an algorithm in the SA type's Cut Process Rule .

Determining your Cut Process Rules can be straightforward or complicated; it depends on how your organization works. Our case study assumes the following:

- For service agreements associated with metered service, we have the following rules:
 - Highest priority. If the customer has life support requirements, create "cut process in light of life support" process.
 - Lowest priority. Otherwise, create a "metered service" cut process.
- For service agreements associated with charitable contributions, create an "expiration only" cut process.

Once you've determined if / how each SA type is cut, you need to design your cut process templates (you'll need one for each unique method of cutting a service agreement). The following table shows the cut process templates referenced above. Adjacent to each process are its events and an indication of when they are triggered.

Cut Process Template	Event Number	Cut Event Type	Dependent On Event(s)	Trigger Date Set To X Days After Completion Of Dependent Events
Metered service	10	Letter - 48 hour disconnect for non-payment warning	N/A - first event	0
	20	Field activity - disconnect for non-payment	10	2
	30	'Service has been disconnected' letter	20	0
	40	Expire service agreement	20	10
Cut process in light of life support	10	Generate delinquent life support customer To Do entry (seeking approval for the cut)	N/A - first event	0
	20	Letter - 72 hour disconnect for non-payment warning	10	0
	30	Generate impending life support cutoff To Do entry to C&C rep	20	3
	40	Field activity - cut for non-payment	30	0
	50	Service has been disconnected letter	40	0
	60	Expire service agreement	40	10
Expiration only	10	Expire service agreement	N/A - first event	

At this point, you've designed the distinct cut process templates. Next, you'll need to design the algorithms that control the lifecycle of their cut processes:

- A template's Cancel Logic algorithm is executed to cancel a process. In addition to cancelling the event, these algorithms are also responsible for inserting [log](#) entry(s).
- A template's Information algorithm is invoked to construct the [override "info string"](#).

Next, extract each unique event type from the above table:

Cut Event Type	Action
48-hour warning letter	Create a customer contact - 48-hour warning letter (letters are created via customer contacts)
72-hour warning letter	Create a customer contact - 72-hour warning letter
Disconnect for non payment	Create a cut for non-payment field activity
Permission to start a cut process for a life support customer	Create To Do seeking permission to send 72 hour letter
Permission to create a cut field activity for a life support cutoff	Create To Do seeking permission to issue a cut field activity
Service has been disconnected letter	Create a customer contact - service cut letter

At this point, you know the distinct event types. Next, you'll need to design the algorithms that control the lifecycle of their events:

- The event type's Event Activation algorithm(s) are executed by the [Overdue / Cut Event Manager](#) on its trigger date. The following points describe the logic embodied in such an algorithm:
 - The activity that happens on the trigger date (e.g., creation of a customer contact, To Do, etc.). Refer to [Cut Events Can Do Many Things](#) for the details.
 - Whether the event is transitioned into the Waiting or Complete state when it's triggered. Refer to [Some Events Can Wait](#) for the details.
 - How the log entry(s) associated with event activation will be constructed. The base-package algorithms allow you to control the verbiage in the log entry by defining the desired message number on the algorithm. This means that you may have to set up new messages. Refer to [Activating Events Should Add A Log Entry](#) for the details.
- The event type's Cancel Logic algorithm(s) are invoked when [an event is cancelled](#). The following points describe the logic embodied in such an algorithm:
 - If the event is allowed to be canceled. This logic may be necessary if some conditions prevent events of this type from canceling. For example, you may want to prevent an event from canceling when there are later dependent events that aren't canceled.
 - Any ancillary actions that take place during cancellation.
 - How the [log entry\(s\)](#) associated with event cancellation will be constructed.
- The event type's Monitor Waiting Event algorithm(s) are invoked to [monitor a waiting event](#). These algorithms are responsible for transitioning a Waiting event to Complete if the object on which it's waiting is complete.
- The event type's Event Information algorithm is invoked to construct the [override "info string"](#).

Set Up Tasks

The above topics provided background information about how overdue processing works. The following discussion summarizes the various set up tasks.

Cut Event Types

You will find that most of the time spent setting up your cut event types is spent setting up the objects that are referenced on the cut event type algorithms. For example, if you use the base-package algorithms, you may need to set up the following:

- The various "types" for the objects created by the plug-ins. For example, if a cut event type creates a To Do entry, you must supply the desired To Do type.
- [Foreign key characteristic types](#) that are used to reference the ancillary objects in the [log entries](#) (e.g., if an event creates a customer contact, the log references this customer contact using a FK characteristic type). Note, many of these will exist in the base-package.
- [Messages](#) that are used to define the verbiage in the [log entries](#). For example, if you use the base-package algorithm that creates a customer contact, you must supply the desired message category and number that contains the verbiage that appears in the log when customer contacts are created. Note, messages have been set up for all base-package algorithms (this means you should not have to set up new messages).
- Etc.

The only way to compile the complete list is to design the parameters for each cut event type algorithm. Refer to [Cut Event Type - Main](#) for the supported plug-in spots.

After you've set up the objects referenced on the algorithms, you can then set up the algorithms. Only then can you set up the cut event types.

Cut Process Templates

After your cut event types exist, you can set up your cut process templates. You will find that most of the time spent setting up your cut process templates is spent setting up the objects that are referenced on the cut process template algorithms. Refer to [Cut Process Template - Main](#) for the supported system events.

SA Type - Cut Process Rules

After you've created your cut process templates, you can set up the algorithms that hold your [cut process rules](#). These are plugged-in on [SA Type - Algorithm](#) in the Cut Process Rule system event.

Overdue Event Types

You will find that most of the time spent setting up your overdue event types is spent setting up the objects that are referenced on the overdue event type algorithms. For example, if you use the base-package algorithms, you will set up the following:

- The various "types" for the objects created by the plug-ins. For example,
 - If an overdue event type creates a To Do entry, you must set up the To Do type.
 - If an overdue event type creates a customer contact, you must set up the customer contact type.
 - If an overdue event type writes off debt, you must set up the adjustment types.
 - Etc.
- [Foreign key characteristic types](#) that are used to reference the ancillary objects in the [log entries](#) (e.g., if an event creates a customer contact, the log references this customer contact using a FK characteristic type). Note, many of these will exist in the base-package.
- [Messages](#) that are used to define the verbiage in the [log entries](#). For example, if you use the base-package algorithm that creates a customer contact, you must supply the desired message category and number that contains the verbiage that appears in the log when customer contacts are created. Note, messages have been set up for all base-package algorithms (this means you should not have to set up new messages).
- Etc.

The only way to compile the complete list is to design the parameters for each overdue event type algorithm. Refer to [Overdue Event Type - Main](#) for the supported plug-in spots.

After you've set up the objects referenced on the algorithms, you can then set up the algorithms. Only then can you set up the overdue event types.

Overdue Process Templates

After your overdue event types exist, you can set up your overdue process templates. You will find that most of the time spent setting up your overdue process templates is spent setting up the objects that are referenced on the overdue process template algorithms. Refer to [Overdue Process Template - Main](#) for the supported system events.

Collection Classes

Set up [collection classes](#) as per your [overdue procedures](#). Make sure to indicate that these collection classes use the Overdue collection method (only accounts linked to collection classes designated as using the Overdue collection method or processed by the Overdue Monitor).

Collection Class Overdue Monitor Rules

After your overdue process templates exist, you can set up your Overdue Monitor Rules. These rules are algorithms plugged in on [Collection Class Overdue Rules](#). You will find most of the time spent setting up these algorithms is spent setting up the objects referenced on the base-package algorithm.

Feature Configuration

You must set up a [Feature Configuration](#) to define parameters that control various overdue processing options.

The following points describe the various **Option Types** that must be defined:

- Trigger Date: Y-Workdays, N-Calendar Days. This option controls how the system computes the trigger dates on overdue and cut events. Enter Y if the system should use workdays. Enter N if the system should use calendar days. Refer to [Calendar vs Work Days](#) for the details.
- Payment Arrangement Type (B/S/A). This option indicates whether your implementation uses the [Balance-Oriented Payment Arrangements](#) (value **S**), [Bill-oriented Payment Arrangements](#) (value **B**) or both (value **A**). The value governs the navigation path for the payment arrangement lines in the [Account History](#) and [Credit & Collection](#) zones.
- Champion Template\$Challenger Template\$Percentage(1-100). You need only set up options of this type if your implementation implements [Champion / Challenger](#) functionality. Options of this type are entered in the format A \$B\$nnn where A is the overdue process template of the champion template, B is the overdue process template of the challenger template, and C is the percent of the time that the system should create the challenger template. The overdue monitor uses this option to override the champion overdue process template X% of the time with the challenger template. You may enter any number of these options (but only one per Champion Template).

Overdue and Cut Event Cancellation Reasons

Overdue events can be cancelled automatically and manually (at the discretion of a user). Regardless of the method of cancellation, a cancellation reason must be supplied. You set up your overdue event cancellation reasons using [Overdue Event Cancellation Reason - Main](#) and [Cut Event Cancellation Reason - Main](#).

Collection Agencies

If you refer debt to collection agents, you must set up your [collection agencies](#).

Alert To Highlight Active Overdue Processes

If you want an alert to appear if the account has active overdue processes, you must configure an appropriate Control Central Alert algorithm ([C1-OD-PROC](#)). This algorithm is plugged in on the [Installation](#) record.

Bill-Oriented Collection - Additional Set Up

The topics in this section provide information on additional set up requirements if you collect on unpaid bills.

One Bill Per Match Event

As mentioned earlier, a bill is considered paid if its financial transactions (FTs) are linked to a balanced match event. To determine a bill's outstanding amount, FTs from different bills cannot be commingled on the same match event (but it's OK for a bill's FTs to be on multiple match events). If you stick by the rule of "just one bill per match event" you will then be able to determine the outstanding balance of a partially paid bill. However, if you mix more than one bill under a match event, a particular bill's balance may become indeterminate.

The following Open-Item algorithm types have been provided by the base package to help enforce this rule:

- The Distribute by Bill Due Date Payment Distribution algorithm ([C1-PYDS-BDU](#)).
- The match by Bill ID Payment Distribution Override algorithm ([C1-PDOV-PYBL](#)).
- The FT cancellation FT Freeze algorithm ([C1-CFTZ-COFT](#)).

If any of your customized plug-ins and processes create match events, it is important that these too enforce this rule. You may want to refer to the base package algorithms as an example of how to do this.

Bill-Based Payment Arrangements

If you set up [payment arrangements for unpaid bills](#), you must configure the Bill-Based Payment Arrangement algorithm and plug it in on your [Collection Class Overdue Rules](#).

It's important to set up the adjustment types used during payment arrangement creation to NOT print on bills. This is because the base-package algorithm will match the adjustments used to transfer debt to the payment arrangement with the adjustment used to reduce the payment arrangement's current amount by the amount of the transfer if all adjustment types are set up to not print.

Bill-Based Write-off

If you [write-off unpaid bills](#), you must set up the following:

- Set up the adjustment type that will be used to write-off an unpaid financial transaction. This adjustment type must be configured as follows:
 - **Adjustment Amount Type** must be Calculated Amount
 - Its distribution code is irrelevant as a separate calculation line will be created for each distribution code on the FT's that is written off and these lines will reference the appropriate distribution code.
 - It must reference an adjustment type char type / value that identifies it as one used to write-off a bill's FTs

- The following algorithms must be defined on the adjustment type:
- The Generate Adjustment system event must reference an algorithm that has the responsibility of determining how to write-off a FT. This algorithm should be determining the FT's GL details and creating a separate adjustment calculation line for each GL detail. The base package is supplied with a sample algorithm that does this ([C1-ADJG-WO](#)).
- The Adjustment Financial Transaction system event should reference an algorithm that impacts current, payoff and the GL by the amount being written off.
- The distribution codes referenced on the financial transactions must be set up with a characteristic that holds the distribution code used to write-off the original amounts. For example,
 - Distribution codes used to record tax liabilities will typically reference the same distribution code for write-off (most organizations reverse tax liabilities at write-off time)
 - Distribution codes used to record revenue will typically reference a write-off distribution code used to record a write-off expense.
 - Distribution codes used to record receivables will typically not reference a write-off distribution code because receivables are implicitly written off when revenue and tax liabilities are written off.
- Set up an adjustment cancellation code used when a users reverses a written-off bill (reversal involves canceling the write off adjustments).
- Set up a Write Off Bill algorithm and plug it in on your [Collection Class Overdue Rules](#). This algorithm will reference the adjustment type described in the previous point.

Alert To Highlight Written Off Bills

If you want an alert to appear if the account has bills with written-off debt, you must configure an appropriate Control Central Alert algorithm ([C1-WO-BILL](#)). This algorithm is plugged in on the [Installation](#) record.

Open-Item Bill Amount Plug-In

You must set up the algorithm that computes the original, unpaid, and write-off amounts of your open-item bills. This algorithm is called by other algorithms when these amounts are needed. This algorithm is plugged-in on [Installation](#) in the Determine Open Item Bill Amounts spot.

Setting Up Overdue Processing

The topics in this section describe how to set up the control tables to implement your overdue processing.

Setting Up Overdue Event Types

An overdue event type encapsulates the business rules that govern a given type of overdue event. Open **Admin > Credit & Collection > Overdue Event Type > Add** to set up overdue event types.

NOTE:

Recommendation. Before using this transaction, we strongly recommend that you review [The Big Picture Of Overdue Events](#).

Description of Page

Enter a unique **Overdue Event Type** code and **Description** for the overdue event type.

Use **Long Description** to provide a more detailed explanation of the purpose of the overdue event type.

The **Algorithms** grid contains algorithms that control important functions. You must define the following for each algorithm:

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Optional / Required	Description
Cancel Logic	Required	This algorithm is executed to cancel an overdue event. Refer to How Are Events Canceled for the details. Click here to see the algorithm types available for this system event.
Event Activation	Required	This algorithm is executed to activate an overdue event on its trigger date. Refer to Overdue Events Can Do Many Things and How and When Events Are Activated for the details. Click here to see the algorithm types available for this system event.
Event Information	Optional - only used if you want to override an overdue event's info string	This algorithm is executed to construct an overdue event's override info string. Refer to Overdue Event Information Is Overridable for the details. Click here to see the algorithm types available for this system event.
Monitor Waiting Events	Optional - only used if events of this type can enter the Waiting state	This algorithm is invoked by the Overdue / Cut Event Manager for events in the Waiting state. Refer to Some Events Wait For Something Before Completing for the details. Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL_OD_EVT_TYPE](#).

Setting Up Overdue Process Templates

An overdue process template encapsulates the business rules that govern a given type of overdue process. Open **Admin > Credit & Collection > Overdue Process Template > Add** to set up overdue process templates.

NOTE:

Recommendation. Before using this transaction, we strongly recommend that you review [The Big Picture Of Overdue Processes](#).

Description of Page

Enter a unique **Overdue Process Template** and **Description** for the overdue process template.

Collecting On Object defines the type of object managed by this overdue process. This field actually references a [foreign key characteristic type](#) that references the managed object. For example, if this overdue process template manages overdue bills, you'd reference a foreign key characteristic that references the bill object.

The **Algorithms** grid contains algorithms that control important functions. You must define the following for each algorithm:

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Optional / Required	Description
Calculate Unpaid & Original Amount	Required	This algorithm is executed to calculate the unpaid and original amounts of the objects associated with the overdue process. These amounts are shown on the overdue process page and in the base-package overdue info string . Click here to see the algorithm types available for this system event.
Cancel Criteria	Required	This algorithm is executed to determine if an overdue process can be cancelled. Refer to How Are Overdue Processes Cancelled for the details. Click here to see the algorithm types available for this system event.
Cancel Logic	Required	This algorithm is executed to cancel an overdue process. Refer to How Are Overdue Processes Cancelled for the details. Click here to see the algorithm types available for this system event.
Hold Event Activation Criteria	Optional - only used if overdue processes of this type can be suspended while some condition is true	This algorithm is executed to determine if the activation of overdue and cut events should be suspended. Refer to Holding Events for the details. Click here to see the algorithm types available for this system event.
Overdue Process Information	Optional - only used if you want to override an overdue process's info string	This algorithm is executed to construct an overdue process's override info string. Refer to Overdue Process Information Is Overridable for the details.

The **Event Types** control the number and type of overdue events linked to an overdue process when it is first created. The information in the scroll defines these events and the date on which they will be triggered. The following fields are required for each event type:

- **Event Sequence.** Sequence controls the order in which the overdue event types appear in the scroll.
- **Overdue Event Type.** Specify the type of overdue event to be created.
- **Days After.** If **Dependent on Other Events** is on, events will be triggered this many days after the completion of the dependent events (specified in the grid). Set this value to 0 (zero) if you want the event triggered immediately after the completion of the dependent events. If **Dependent on Other Events** is off, events will be triggered this many days after the creation of the overdue process. Refer to [How and When Events Are Activated](#) for the details.
- If **Dependent on Other Events** is on, define the events that must be completed or cancelled before the event will be triggered.
 - **Sequence** is system-assigned and cannot be specified or changed.
 - **Dependent on Sequence** is the sequence of the dependent event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_OD_PROC_TMP](#).

Setting Up Cut Event Types

A cut event type encapsulates the business rules that govern a given type of cut event. Open **Admin > Cut Event Type > Add** to set up cut event types.

NOTE:

Recommendation. Before using this transaction, we strongly recommend that you review [Cut Events Are Like Overdue Events](#).

Description of Page

Enter a unique **Cut Event Type** code and **Description** for the cut event type.

Use **Long Description** to provide a more detailed explanation of the purpose of the cut event type.

The **Algorithms** grid contains algorithms that control important functions. You must define the following for each algorithm:

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Cancel Logic	Required	This algorithm is executed to cancel a cut event. Refer to How Are Events Canceled for the details. Click here to see the algorithm types available for this system event.

Event Activation	Required	This algorithm is executed to activate a cut event on its trigger date. How and When Events Are Activated for the details. Click here to see the algorithm types available for this system event.
Event Information	Optional - only used if you want to override a cut event's info string	This algorithm is executed to construct a cut event's override info string. Refer to Cut Event Information Is Overridable for the details. Click here to see the algorithm types available for this system event.
Monitor Waiting Events	Optional - only used if events of this type can enter the Waiting state	This algorithm is invoked by the Cut / Cut Event Manager for events in the Waiting state. Refer to Some Events Wait For Something Before Completing for the details. Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CUT_EVT_TYPE](#).

Setting Up Cut Process Templates

A cut process template encapsulates the business rules that govern a given type of cut process. Open **Admin > Cut Process Template > Add** to set up cut process templates.

NOTE:

Recommendation. Before using this transaction, we strongly recommend that you review [The Big Picture Of Cut Processes](#).

Description of Page

Enter a unique **Cut Process Template** and **Description** for the cut process template.

The **Algorithms** grid contains algorithms that control important functions. You must define the following for each algorithm:

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Cancel Logic	Required	This algorithm is executed to cancel a cut process. Refer to How Are Cut Processes Cancelled for the details. Click here to see the algorithm types available for this system event.

Cut Process Information

Optional - only used if you want to override a cut process's info string

This algorithm is executed to construct a cut process's override info string. Refer to [Cut Process Information Is Overridable](#) for the details.

Click [here](#) to see the algorithm types available for this system event.

The **Event Types** control the number and type of events linked to a cut process when it is first created. The information in the scroll defines these events and the date on which they will be triggered. The following fields are required for each event type:

- **Event Sequence.** Sequence controls the order in which the cut event types appear in the scroll.
- **Cut Event Type.** Specify the type of cut event to be created.
- **Days After.** If **Dependent on Other Events** is on, events will be triggered this many days after the completion of the dependent events (specified in the grid). Set this value to 0 (zero) if you want the event triggered immediately after the completion of the dependent events. If **Dependent on Other Events** is off, events will be triggered this many days after the creation of the cut process. Refer to [How and When Events Are Activated](#) for the details.
- If **Dependent on Other Events** is on, define the events that must be completed or cancelled before the event will be triggered.
 - **Sequence** is system-assigned and cannot be specified or changed.
 - **Dependent on Sequence** is the sequence of the dependent event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CUT_PROC_TMP](#).

Setting Up Collection Class Overdue Rules

Collection class overdue rules contain algorithms that impact accounts associated with a given collection class, division and currency code are managed. Open **Admin > Collection Class Overdue Rules > Add** to set up collection class overdue rules.

NOTE:

Recommendation. Before using this transaction, we strongly recommend that you review [Different Overdue Rules For Different Customers](#).

Description of Page

Enter the **Collection Class**, **CIS Division** and **Currency Code** to which the rules apply.

The **Algorithms** grid contains algorithms that control important functions. You must define the following for each algorithm:

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
--------------	---------------------	-------------

Bill-Based Payment Arrangement	Optional - only specified if your implementation uses bill-oriented payment arrangements	This algorithm is executed to handle the creation, breaking and canceling of a Bill-Oriented Payment Arrangements . Click here to see the algorithm types available for this system event.
Overdue Monitor Rule	Required	This algorithm is invoked by the Overdue Monitor to analyze an account's debt. Refer to How Does The Overdue Monitor Work for the details. If you have multiple rules (and therefore multiple algorithms), please take care when assigning the sequence number, as the Overdue Monitor will invoke these rules in sequence order. Click here to see the algorithm types available for this system event.
Write Off Bill	Option - only specified if your implementation writes-off bills	This algorithm is executed to handle the write-off and write-off reversal of a bill. Refer to Writing Off Bills . Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_OD_RULE_ALG](#).

Setting Up Overdue Event Cancellation Reasons

An overdue event cancel reason must be supplied before an overdue event can be canceled. Open **Admin > Credit & Collection > Overdue Event Cancel Reason** to define overdue event cancellation reasons.

Description of Page

Enter an easily recognizable **Overdue Event Cancel Reason** and **Description** for each cancellation reason.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_OEVT_CAN_RSN](#).

Setting Up Cut Event Cancellation Reasons

A cut event cancel reason must be supplied before a cut event can be canceled. Open **Admin > Cut Event Cancel Reason** to define cut event cancellation reasons.

Description of Page

Enter an easily recognizable **Cut Event Cancel Reason** and **Description** for each cancellation reason.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_C EVT_CAN_RSN](#).

Defining Interval Billing Options

NOTE:

The transactions described in this section are only available if the interval billing module is turned on. This Customer Care and Billing functionality has essentially been deprecated and should be performed within a Meter Data Management system. This functional area remains within Customer Care and Billing exclusively for upgrading customers who currently use Interval Billing within an earlier version of this system.

NOTE:

The interval rating and billing functionality described in this chapter was designed and developed prior to the big data requirements of the smart grid. Implementations should use a meter data management application, such as Oracle Utilities Meter Data Management, for smart grid requirements.

The system provides capability to bill complex customers who measure usage in small intervals, such as one hour, thirty minutes, fifteen minutes, etc. The "Interval Billing" subsystem covers a variety of functionality including:

- Capturing billable interval data quantities linked to a service agreement
 - Capturing "raw" interval data quantities linked to service points, which may be later aggregated to create billable quantities for the service agreement
 - Application of contract-based modifications to produce interval data quantities
 - Apply interval prices to interval quantities. (Sometimes this is known as "direct billing")
 - Support time of use mapping, including override maps created for certain special periods, such as interruptions
 - Support time of use pricing and time of use contract values
 - Support special contract options under which certain calculations supporting a contract's rate may be overridden or altered occasionally for specific periods of time
-

NOTE:

The ability to capture and maintain interval prices and TOU prices is officially part of the Rates subsystem. Although we will refer to them here, the main documentation is in the Rates administration guide.

WARNING:

Setting up the interval billing control tables is as challenging as your organization's interval billing rules. If you have simple rules then your setup process will be straightforward. If your interval billing rules are complicated (e.g., specific prices for each customer, complicated TOU Mapping rules etc.), then your setup process will be more challenging.

NOTE:

Separate modules. Interval billing functionality is associated with separate modules, Complex Billing and Meter Data Management. If these modules are not applicable to your business you may turn them off. Refer to [Turn Off A Function Module](#) for more information.

Interval Billing Table Setup Sequence

The following table defines the table setup sequence required if your company has purchased the interval billing component.

Function	Path
General Environment	
Seasonal Time Shift	Admin, Seasonal Time Shift
Time Zone	Admin, Time Zone
Installation - Framework	Admin, Installation Options - Framework Indicate whether Seasonal Time Shift is required.
Installation	Admin, Installation Options Set Base Time and Start Day Option.
Interval Billing Environment	
Bill Factor	Menu, Rates, Bill Factor Note: earlier, you may have created bill factors for your general Rates environment. At this point, you may need to add more bill factors to satisfy your interval billing needs.
Interval Profile Relationship Type	Admin, Interval Profile Relationship Type [Note - you won't be able to define the collection of valid Interval Profile Types until after you define the Interval Profile Types.]
Interval Profile Type	Admin, Interval Profile Type [Note - you may need to define new algorithm types and algorithms if your interval profile type requires creation or validation algorithms.]
Algorithm	Admin, Algorithm. You will need to set up the creation and validation algorithms needed for an Interval Profile Type.
Shared Profiles	Menu, Interval Billing, Interval Profile Note: this is needed at this time if you want to create Start Options, which reference shared profiles.
Interval Registers	
Interval Register Type	Admin, Interval Register Type
Meter Configuration Type	Admin, Meter Configuration Type Note: earlier, you may have created meter configuration types for your general meter environment. At this point, you may need to add more meter configuration types for registers, which are used for interval or index channels.
Meter Type	Admin, Meter Type Note: earlier, you may have created meter types for your general meter environment. At this point, you may need to add more meter types for meters, which are used for interval or index channels.
Time of Use Billing	
Time of Use	Admin, Time of Use Note: earlier, you may have created time of use codes for your meter environment. At this point, you may need to add more time of use codes to satisfy your time of use mapping.

TOU Group	Admin, TOU Group
Bill Factor	Menu, Rates, Bill Factor At this point you may need to set up bill factors that are specifically for time of use pricing.
TOU Map Relationship Type	Admin, TOU Map Relationship Type [Note - you won't be able to define the collection of valid TOU Map Types until after you define the TOU Map Types.]
TOU Map Type	Admin, TOU Map Type
Algorithm	Admin, Algorithm. You will need to set up any creation algorithms needed for your TOU map types.
TOU Map Templates	Admin, TOU Map Template Note: this is not required, but will help to set up data for your TOU Maps.
Shared TOU Maps	Menu, Interval Billing, TOU Map Note: this is needed at this time if you want to create Start Options, which reference shared TOU maps.
Contract Options	
Contract Option Type	Admin, Contract Option Type
Algorithm	Admin, Algorithm. You will need to set up any validation algorithms needed for your contract option types.
Contract Option Event Type	Admin, Contract Option Event Type
SA Interval Billing Rate Environment	
Rate	At this point, you are ready to set up your interval billing and time of use rates and calculation rules. Refer to the Rate Environment section in the Control Table Setup Sequence .
SA Interval Billing Controls	
SA Type	Admin, SA Type Note: earlier you may have created your SA Types. At this point you may need to modify interval related SA Types to add valid interval information. Refer to the SA Type section in the Control Table Setup Sequence .
Start Option	Admin, SA Type Start Option Note: earlier you may have created your SA Type Start Options. At this point you may need to modify interval related SA Type Start Options to add valid interval information. Refer to the SA Type section in the Control Table Setup Sequence .

NOTE:

You may have customers with interval billing, time of use billing and contract options all required for their rate. For simplification of the table, these control tables were listed in separate sections.

The Big Picture of Interval Billing

This section provides an overview of important Interval Billing concepts with which you should be familiar before you set up your Interval Billing control tables.

Interval Pricing

This section provides an overview of concepts related to setting up interval pricing options for your rates. Applying interval prices to interval quantities is sometimes referred to as 'direct billing'.

Interval Pricing Rate Application

Interval Pricing is the term used to describe applying interval prices to interval quantities to arrive at a bill calculation line item.



What data is needed in order to apply a rate component for an interval pricing scenario and how is this data defined?

- You need prices that vary at a given interval. Interval prices are stored for a **Bill Factor**/characteristic. Refer to [Bill Factor Interval Values](#) for more information.
- You need consumption values for each corresponding interval. The consumption values are stored for an Interval Profile linked to a service agreement. Refer to [Billable Interval Quantities for a Service Agreement](#) for more information. In order to find the correct interval profile for the service agreement, the rate component will reference a **Profile Relationship Type**. Refer to [Physical Attributes of Interval Data vs. Its Role](#) and [Interval Data Serves a Role for a Service Agreement](#) for more information.
- You need an algorithm in order to know how to apply the prices correctly. The rate component will reference a **Calculation Algorithm**, which will be executed to apply the prices to the quantities.

Refer to [Designing Your Interval Rate Components](#) for help in designing rate components of this type. Refer to [Setting Up Interval Pricing Rate Components](#) for more information about setting up this type of rate component.

Physical Attributes of Interval Data vs. Its Role

You will see, as you learn more about the design of interval billing, that there are two control tables that are important for defining billable interval data:

- Profile Type - this defines the physical attributes of the interval data

- Profile Relationship Type - this defines the role that the interval data is playing for a particular contract. You can also think of this as defining the business purpose of the data.

Interval Data Physical Attributes

When defining a collection of billable data, there are basic attributes, which need to be defined:

- Unit of Measure
- Minutes per Interval
- Service Quantity Identifier
- Is this data owned by a service agreement or is it common data?

These are physical attributes of the data and have nothing to do with the business purpose of the data. This information is defined on the Profile Type.

The profile type may also include algorithms related to its data:

- Validation algorithms may be used to check and correct various conditions related to the interval data. Refer to [Validation of Profile Data](#) for more information.
- A creation algorithm may be used to derive data for a profile. Refer to [Creation of Profile Data through Data Derivation](#) for more information.

Refer to [Designing Interval Profile Types](#) for more information.

You will see later that classic TOU Maps also have a classic TOU Map Type. Refer to [Physical Attributes of a classic TOU Map](#) for more information.

Business Role of Interval Data

The Profile Relationship Type is used by the system to indicate the "role" that a collection of billable interval data is playing. The following are some examples of roles that interval data may play:

- Measured Demand
- Contract Demand
- Aggregated Heating Demand
- Hedge Cover
- Excess Demand
- Reactive Energy

The profile relationship type is used by the rate component to indicate the data being billed. For example, the rate is billing "excess demand". When applying the rate, the system will determine which data is playing the role of "excess demand" for the service agreement. Refer to [Billable Interval Quantities for a Service Agreement](#) and [Interval Data Serves a Role for a Service Agreement](#) for more information.

You will see later that classic TOU Maps also have a "role". Refer to [Business Role of a TOU Map](#) for more information.

The Business Role Defines Interval Data Physical Attributes

To enable proper setup, you will need to define the valid [profile types](#) for each profile relationship type.

Raw Data Collection and Aggregation

This section provides an overview of concepts related to setting up your control tables to support the capturing of raw interval data.

FASTPATH:

Refer to [The Big Picture of Raw Data Collection and Aggregation](#) for more information.

NOTE:

If your company uses an external system for collecting, adjusting and aggregating raw interval data, then you may skip this section.

Physical Attributes of Raw Interval Data

Raw interval data is available for use by algorithms in the system to create billable data for the service agreement. This data will not have any 'business' role, as with the billable interval data, described above.

As a result only 'physical' attributes of the raw interval data need to be defined, such as:

- Unit of Measure
- Minutes per Interval
- Service Quantity Identifier

This information is defined on the Interval Register Type.

The [interval register type](#) may also include algorithms related to its data:

- Validation algorithms may be used to check and correct various conditions related to the interval data. Refer to [Validation of Register Data](#) for more information.

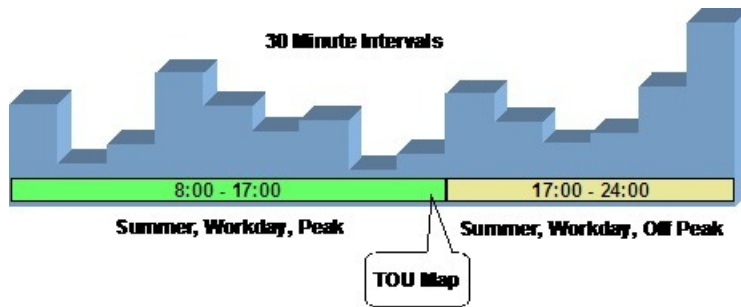
Time of Use Billing

This section provides an overview of concepts related to setting up your control tables to support time of use billing.

Defining Time of Use Periods

Many customers choose not to price their interval data using interval prices. Time of Use Mapping enables a customer to map out time of use periods for their usage. This option for interval data might be preferred because:

- Typically it involves fixed prices for the use periods
- It is more manageable than direct billing
- It is easier for a customer to forecast and budget



A classic TOU Map holds the collection of time period definitions. The TOU Map has a TOU Map Type, which defines the minutes per interval. This is similar to the interval profile and profile type.

The time period definitions for a classic TOU Map indicate the TOU code for a given date and time.

Map #123 (TOU Group 2)	
Effective 1 Jan 2000	
Interval Date/Time	
30/Apr/00 16:30	On Peak/Winter
30/Apr/00 16:45	On Peak/Winter
30/Apr/00 17:00	Off Peak/Winter
30/Apr/00 17:15	Off Peak/Winter
---	---
01/May/00 7:45	Off Peak/Summer
02/May/00 8:00	On Peak/Summer
02/May/00 8:15	On Peak/Summer
---	---

All the possible time of use codes for a given map are grouped together in a TOU group. Refer to [Grouping of TOU Codes](#) for more information.

Grouping of TOU Codes

Overview

The time of use map's purpose is to define time of use codes for given time periods. For example:

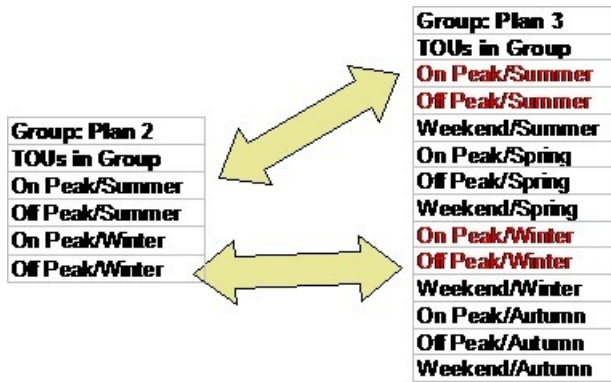
1/Jan/2001 05:00:00 is Off Peak, Winter

1/July/2001 13:00:00 is Weekend, Summer

5/July/2001 12:30:00 is On Peak, Summer

The time of use codes of "Off Peak, Winter" and "Weekend, Summer" are user-defined and use the same [TOU code](#) available for defining TOU for simple metering.

The set of time of use codes that make up a certain classic TOU map, are grouped together using the TOU group. The TOU group is a logical grouping of time of use codes. A TOU can exist on more than one TOU group.



The [TOU group](#) is used to define the collection of time of use codes available for a classic TOU map. In addition, you will see TOU group used in other areas of interval billing where a collection of time of use codes is required.

TOU Sequence

If desired, you may use sequence number to indicate the relative position or relative priority of each TOU code within a TOU group. This sequence number is not used by any system functionality, but is available for you to use in a plug-in algorithm.

For example, assume that your customer's contract states that if the usage for the time period "ON" is below a certain contract limit, they will be charged the "ON" price. However, if the usage exceeds this contract limit, they will be charged a higher price for this usage. Let's assume the same is true for "OFF". You may choose to implement this as follows:

- Besides TOU codes of "ON" and "OFF", create additional TOU codes called "ONEXCESS" and "OFFEXCESS"
- Define the TOU codes within the TOU group with sequences values as follows:

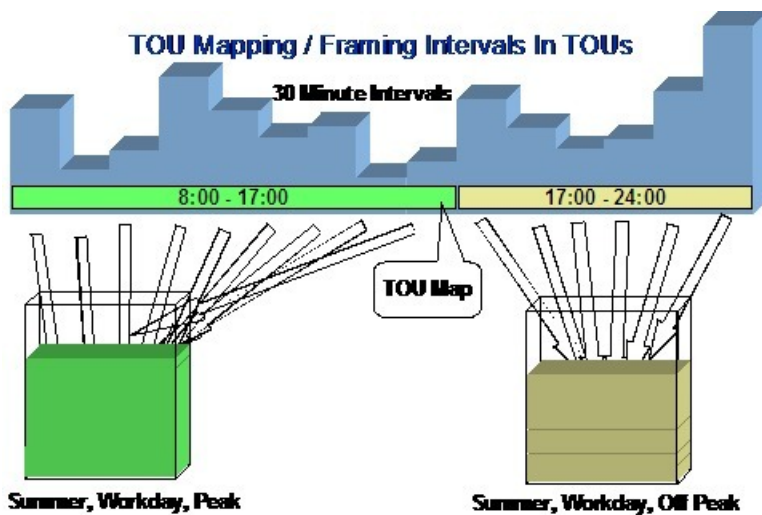
Time of Use	TOU Sequence
ON	1
ONEXCESS	2
OFF	3
OFFEXCESS	4

- Your classic TOU map will only define time periods for ON and OFF, but you will have TOU prices for all four time of use values.
- Design your TOU pricing algorithm so that if the usage for a given TOU is within the contract limit, the price for this TOU is used. However, if the usage exceeds the contract limit, then it will find the TOU with the next highest sequence and use the price for that TOU.

Time of Use Mapping and Pricing

The following section describes the logic used by the system to map interval quantities to time of use codes using a TOU map, and subsequently apply prices to these mapped quantities.

At some point during billing, the system will take interval quantities linked to the service agreement and map them to time of use quantities based on a TOU Map.



For simple billing, this type of data manipulation is typically handled using an SQ rule. Keep in mind that SQ Rules are processed before applying the rate components and therefore they do not have knowledge of system breaks that may cause price proration.

Proration occurs when a given price covers a period smaller than the billing period AND the system does not have readings for the period covered by the price. When this occurs, the system will prorate the usage to apply the correct price. Refer to [Effective Dates & Price Proration](#) for more information about proration.

If you read carefully the price proration information, you'll see that if the service quantities are peak quantities, the quantities are not prorated but the charges are prorated. For other cases, the system prorates the service quantities.

When your interval quantities are peak quantities, the same should apply. The mapping should occur prior to price breaks, for example, using an SQ rule. Then, the TOU pricing rate components would apply prices to the mapped SQ quantities.

For non-peak interval quantities, readings exist for any price break situation. As a result, no service quantity proration is necessary. The mapping simply needs to occur after all price breaks are determined - at the rate component level.

TOU Pricing Rate Component

A special rate component exists to perform time of use mapping and time of use pricing. In fact, this rate component uses a calculation algorithm, so the algorithm can be written to perform whatever logic you need it to perform. The system is shipped with an algorithm, which performs mapping only and another algorithm, which may perform mapping and pricing, or just pricing.

- The mapping algorithms populate the read details collection by default because this collection can contain the quantities for each UOM/TOU/SQI along with a date range (This is important for [price breaks](#).) In addition, you may configure the algorithm to also produce SQ quantities, which would represent the total quantities for each UOM/TOU/SQI.
- The pricing algorithm may be configured to apply prices to the SQ collection (for measures peak scenarios) or the read details collection (for non-measures peak scenarios).

Refer to [Designing Your Time of Use Rate Components](#) for help in designing rate components of this type. Refer to [Setting Up TOU Pricing Rate Components](#) for more information about setting up this type of rate component.

The TOU pricing rate component may reference either a [TOU bill factor](#), which contains prices that differ for each time of use code or a regular bill factor, which will contain a single price to apply to all mapped quantities.

Classic TOU Map Used For Mapping & Pricing

The classic TOU map used by the rate component for mapping is defined either directly on the rate component or it is linked to the service agreement.

- If ALL service agreements linked to the rate component's rate should use the same classic TOU map, then link the classic TOU Map to the rate component
- If some service agreements use different TOU maps than others, the TOU map must be linked to the service agreement. In this case, you must indicate a TOU Map Relationship Type on the rate component. The TOU map relationship type defines the [business role of TOU map](#).

Refer to [Sharing TOU Map Data](#) for more information.

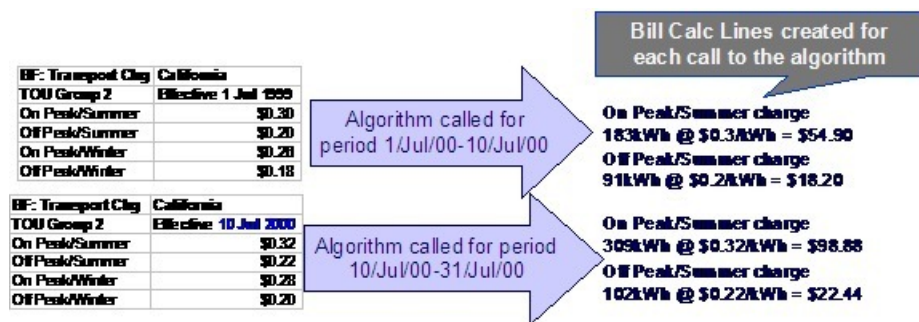
TOU Price Breaks

This section describes functionality related to price changes for time of use bill factor values. This logic is only applicable when the interval quantities do not measure peak quantities.

- [Algorithms are Called for Each TOU Price Break](#)
- [Algorithms Are Responsible for Storing Calculation Details](#)

Algorithms are Called for Each TOU Price Break

Rate application logic understands how to determine price breaks for the bill factor referenced on your rate component and calls the calculation algorithm for each price break. The TOU pricing algorithm will be passed the appropriate date/time range to handle mapping and pricing and will produce separated bill calculation lines for each time period.



NOTE:

Single TOU Group. If your rate component references a TOU bill factor, the price breaks for the bill factor are determined using a single TOU group. If multiple classic TOU maps are applicable for the billing period and each refers to a different TOU group (via its [TOU map type](#)), the system uses the SA type's rate selection date to select the appropriate TOU group to use to determine the collection of prices to apply.

[Top of Page](#)

Algorithms Are Responsible for Storing Calculation Details

When the TOU mapping algorithms calculate the mapped quantities for a given date/time range, the results are stored in the [calculation/audit read details](#) collection on the bill segment. This collection enables the system to record the mapped quantities along with period represented by each quantity.

Your algorithms may opt to additionally capture the total quantity for each time of use in the SQ Details collection as an audit.

Read Details

OnSum	01/Jul/00-10/Jul/00	183kWh
OffSum	01/Jul/00-10/Jul/00	91kWh
OnSum	10/Jul/00-31/Jul/00	309kWh
OffSum	10/Jul/00-31/Jul/00	102kWh

Algorithms use the read detail collection to store interim UOM/TOU/SQI details with date and time

SQ Details

OnSum	492kWh
OffSum	193kWh

The base algorithms also provide the option of populating the SQ collection with the total quantity for each UOM/TOU/SQI

NOTE:

Bill Factor prices have effective dates only. There is no effective time. When a price break occurs, the algorithm will determine the time for the price breaks in the same manner that billing time is determined. Refer to [Start and End Times for Billing](#) for more information.

The system provides one sample algorithm [RCTMS](#) that performs TOU mapping and another algorithm [RCTMPS](#) that performs TOU mapping and pricing or pricing alone. These algorithms support TOU price break logic as follows:

- The mapping functions for both algorithms populate the calculation/audit read details collection and allow you to set a parameter to indicate whether SQ entries should also be created.
- Based on a parameter, the pricing portion of RCTMPS applies the appropriate TOU prices to the mapped quantities stored in the calculation/audit read details collection.

If RCTMPS has been configured to perform mapping and pricing, it will perform the two functions as separate steps: first it maps the quantities and populates the read details, then it applies TOU prices to the read details. It does this for the following reasons:

- You may have a rate that requires manipulation of mapped quantities prior to applying prices. For example, perhaps you must add service quantities from a non-interval meter to your mapped quantities prior to applying prices.
- You may need to [override the mapped quantities](#) when recalculating a bill. The base algorithms will not re-map the quantities if the details were overridden. The pricing portion applies the prices to the overridden quantities.

Overriding Mapped Quantities

The mapping algorithms provided with the base product record the mapped quantities in the [calculation/audit read details](#) collection on the bill segment. When regenerating a bill segment, it is possible that circumstance requires you to override the details in this collection rather than allowing the system to map the interval quantities again.

Refer to [How To Override Service Quantities On A Specific Bill Segment](#) for more information.

The pricing functionality in the algorithms provided by the base product detect when mapped quantities have been overridden and apply the TOU prices to the overridden quantities.

Physical Attributes of TOU Map vs. Its Role

You will see, as you learn more about the design of TOU mapping, that there are two control tables that are important for defining TOU maps:

- TOU Map Type - this defines the physical attributes of the TOU Map
- TOU Map Relationship Type - this is needed for TOU maps that are linked to a service agreement. It defines the role that the TOU Map is playing for that service agreement. You can also think of this as defining the business purpose of the data.

Physical Attributes of a Classic TOU Map

When defining a classic TOU map, there are basic attributes, which need to be defined:

- The collection of possible time of use codes to which the data will be mapped. These are grouped together into a [TOU group](#).
- Minutes per Interval
- Is this data owned by a service agreement or is it common data?

These are physical attributes of the data and have nothing to do with the business purpose of the data. This information is defined on the classic TOU Map Type.

The [classic TOU map type](#) may also include an algorithm used to [automatically create data for the classic TOU map](#).

This is the same model used for interval profile data. Refer to [Physical Attributes of Interval Data](#) for more information.

Business Role of a Classic TOU Map

The TOU Map Relationship Type is used for TOU maps that are linked to the service agreement. It indicates the "role" that a classic TOU Map is playing for a particular service agreement. The following are some examples of roles that a classic TOU Map may play:

- Map for Measured Demand
- Map for Contract Demand
- Map for Excess Demand
- Map for Reactive Energy

If your rate component requires a classic TOU map but the TOU map is not common to all service agreements for the rate, then the rate component must indicate a TOU Map Relationship Type. The rate component algorithm uses this information to find the appropriate classic TOU map linked to the service agreement. Refer to [Classic TOU Maps Serve a Role for a Service Agreement](#) for more information.

This is the same model used for interval profile data. Refer to [Business Role of Interval Data](#) for more information.

The Business Role Defines Valid Physical Attributes for a Classic TOU Map

To enable proper setup, you will need to define the valid [classic TOU map types](#) for each TOU map relationship type.

FASTPATH:

For more information about defining classic TOU maps and recording the time period data, refer to [Time of Use Mapping Background Topics](#).

Designing Interval Billing Options

Your interval billing options control how interval data may be maintained and billed for your customers with this capability.

FASTPATH:

For more information about interval billing, see [The Big Picture Of Interval Billing](#).

WARNING:

There are many ways to design your interval billing options. The flexibility of the system may add to the challenge of determining the best way to set up your control tables. In this section, we provide information to help you understand the ramifications of the various options. Before you set up your production data, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

The design of your interval billing options is an iterative process. Over time, you will develop intuitive skills that will allow you to skip some iterations. We recommend using the various steps in this section as a guide. When you are finished with this guide, you will be able to set up your interval billing options.

Designing Your Interval Billing Rate Options

This section assumes that you are familiar with the Rates subsystem and especially with the section [Understanding Calculation Groups and Rules](#). Your interval billing rate components will likely contain a combination of standard rate components and interval billing rate components. The focus in this section will be on designing rate components for interval billing and time of use mapping.

Designing Your Interval Rate Components

Although your rates will likely contain a combination of interval pricing and time of use pricing, we have separated these topics with respect to walking you through the control table setup.

To set up the rate components that support interval billing, we recommend using the following table as your guide.

RC Type	Bill Factor	Interval Profile Relationship Type	Algorithm	TOU Map Relationship Type

Obtain copies of existing bills that use the rate in question. If the rate is new, then write up EXACTLY how the information should appear on the customers' printed bills.

- Next, try to identify the components of the bill that are related to interval billing.
 - Are any of the lines produced as a result of direct billing? In other words, were interval prices applied to interval quantities?
 - Are any of the lines produced using interval quantities applied to a fixed price?

Start filling out the table with descriptions of what is needed to produce each line.

- Any line that is produced as a result of taking interval quantities and applying prices will be defined with the "Interval Pricing" rate component type.
 - This rate component type will require an Interval Profile Relationship Type (which is used to define the source data to price).
 - A bill factor will need to be defined. This bill factor may contain interval quantities or simple values.
 - An algorithm, which knows how to apply the prices to the quantities, is needed. The system provides the following rate component algorithms as examples.

- Rate Components related to time of use mapping and pricing are discussed below. Refer to [Designing Your Time Of Use Rate Components](#).

RC Type	Bill Factor	Interval Profile Relationship Type	Algorithm	TOU Map Relationship Type
Interval Pricing	Strike Price	Hedge Cover	Apply prices to quantities assuming a continuous curve	N/a
Interval Pricing	Spot Market Prices	Amount in Excess of Hedge Cover	Apply prices only to positive quantities assuming a continuous curve	N/a

Designing Your Billing Factors

To apply prices to interval quantities, bill factors are recommended. In fact, to be able to handle interval prices, you will need to use bill factors.

FASTPATH:

To design and set up your bill factors, refer to [Setting Up Bill Factors](#) for more information.

Designing Your Interval Billing Controls

Designing Interval Profile Relationship Types

Recall that the interval profile relationship type can be thought of as the role that interval data will play. You can also think of it as the business description of the interval data.

The starting point for designing your interval profile relationship types is the rate. First, identify the relationship types that your rate needs in order to produce a bill

Using the above rate as an example, two profile relationship types have been identified

Profile Relationship Type	Description
HEDGECOVER	Hedge cover
OVRHEDGE	Amount over hedge cover.

Now, you must think about what other relationship types are required to produce the above relationship types needed by rates. In our above example, we will need to define a profile relationship type whose data is compared to the hedge cover data to produce the OVRHEDGE data. Let's assume that this new profile relationship type is related to measured demand.

Profile Relationship Type	Description
HEDGECOVER	Hedge cover
OVRHEDGE	Amount over hedge cover.

In this manner, you will be able to successfully define your profile relationship types.

Designing Interval Profile Types

Now that you have your profile relationship types defined, you need to begin defining profile types. The interval profile type defines the physical attributes of the interval data.

The essential attributes of any profile type are the UOM/SQI, the minutes per interval and the associated algorithms. The easiest way to start defining your interval profile types is to start with the profile relationship type and determine what type of data each customer may have for the same profile relationship type.

Let's start with our profile relationship types and make some assumptions regarding the physical attributes of the data that a customer may have for each of these roles:

- Let's assume that a customer's demand may be measured in either 30-minute intervals or 60-minute intervals. In either case, the UOM is kW and SQI is not applicable. Let's also assume that the measured data may be interfaced from an external source or may be aggregated from interval register data.
- Let's assume that the hedge cover is common, but that these values may also be in 30-minute intervals or 60-minute intervals.
- Finally, let's assume that our spot market prices are at 60-minute intervals and that the algorithm, which rates uses to apply the interval prices, expects the "amount in excess of the hedge cover" to be in 60-minute intervals. This means that we need two different algorithms for our OVRHEDGE data. One algorithm is used for a customer with 60-minute intervals and simply subtracts the two curves and produces the resulting data. The second algorithm is used for a customer with 30-minute intervals. It performs two steps. It subtracts the hedge from the measured demand and adds together each resulting 30-minute pair to produce a 60-minute OVRHEDGE curve.

We will assume that all of our profile data will follow the same seasonal time shift as our base time zone. Refer to [Time Zone and Time Changes](#) and [Designing Your Time Options](#) for more information.

Prof. Rel. Type	Profile Type	Common/SA Owned	UOM/ SQI	Min/ Interval	Seasonal Time Shift	Algorithm
MEASDMD	DMDKW60	SA Owned	KW	60	USShift	N/a
	DMDKW30	SA Owned	KW	30	USShift	N/a
	DMDKW60AGG	SA Owned	KW	60	USShift	Aggregate interval register data
	DMDKW30AGG	SA Owned	KW	30	USShift	Aggregate interval register data
HEDGECVR	HEDGE60	Common	KW/ HG	60	USShift	N/a
	HEDGE30	Common	KW/ HG	30	USShift	N/a
OVRHEDGE	OVRHG60	SA Owned	KW/ OVR	60	USShift	Compare MEASDMD to HEDGECVR to produce 60-minute over hedge curve

OVRHG30	SA Owned	KW/ OVR	30	USShift	Compare MEASDMD to HEDGEVCR and add 30-min interval pairs to produce 60-min 'over-hedge' curve
---------	----------	---------	----	---------	--

NOTE:

The above table accomplishes two steps: defining profile types and defining the valid profile types for a profile relationship type. When setting up this data, you will need to define your profile types first and then link them to the appropriate profile relationship type.

As with most of your control table design, this is an iterative process. As you design your profile types, you may see the need for new profile relationship types. You may find that a profile type will be valid for more than one profile relationship type.

FASTPATH:

During your definition of the profile types, you may determine that new UOMs need to be defined. Refer to [Setting Up Unit Of Measure Codes](#) for more information about defining units of measure.

During your definition of the profile types, you may determine that new pre-processing calculation groups that need to be defined. Refer to [Understanding Calculation Groups and Rules](#) for more information about defining SQI values.

NOTE:

Now that you have designed the control table values required to support your rate, we recommend that you set up [start options](#) for your SA types to assist a CSR in setting up a customer for this rate.

All of the above steps will need to be repeated for each interval billing rate that your company offers. For each rate, you need to define your rate components, your billing factors, your profile relationship types and then the appropriate profile types.

Designing Your Raw Data Options

NOTE:

If your company uses an external system for collecting, adjusting and aggregating raw interval data, then you may skip this section.

The term "Channel" is often used for devices that may store interval data. A physical channel may hold interval data or index readings.

- Interval channels will contain collections of interval data
- Index channels are a collection of time of use registers and readings for these registers are no different than standard register readings

In either case, a "channel" is represented in the system by a register and a meter will represent a logical grouping of channels (index or interval). Refer to [The Structure Of A Meter](#) for more information.

Designing Your Interval Register Types

Each register, which will record raw interval data will require an interval register type.

The essential attributes of an interval register type are the UOM/SQI, the minutes per interval and any validation algorithms.

- For our examples, let's assume that raw interval data may be measured in kilowatts or megawatts and may be measured in either 30-minute intervals or 60-minute intervals.
- For our examples, let's assume that the data may be standard recorded data or it may be calculated excess data. We will use an SQI to further label the excess demand.
- For each register type in our example, we will use a validation algorithm, which will verify that the intervals are correct according to the interval size on the interval register type.

We will assume that all of our raw interval data will follow the same seasonal time shift as our base time zone. Refer to [Time Zone and Time Changes](#) and [Designing Your Time Options](#) for more information.

Interval Register Type	UOM/ SQI	Min/ Interval	Seasonal Time Shift	Validation Algorithm
KW60	KW	60	USShift	Validate interval size
KW60EXC	KW/EXCESS	60	USShift	Validate interval size
KW30	KW	30	USShift	Validate interval size
KW30EXC	KW/EXCESS	30	USShift	Validate interval size
MW60	MW	60	USShift	Validate interval size
MW60EXC	MW/EXCESS	60	USShift	Validate interval size
MW30	MW	30	USShift	Validate interval size
MW30EXC	MW/EXCESS	30	USShift	Validate interval size

We expect that you will need more algorithms than we supply. Your algorithms will be based on any number of factors. Be aware that new algorithms may require programming. See [How To Add A New Algorithm](#) for more information.

Designing Your Raw Data TOU Groups

In addition to designing TOU groups to be used by time of use maps, you may also set up [TOU groups](#) to define a valid collection of time of use values for a given meter configuration.

Designing Your Meter Configuration Types

You must define a meter configuration type for each valid collection of interval and index registers. For meter configuration types with index registers, you may indicate a TOU group, which contains the valid collection of time of use codes for the registers. For each interval register, you must check the Interval switch and indicate the appropriate Interval Register Type. A meter configuration may contain

- A single interval register
- Multiple interval registers
- A collection of index registers
- A combination of interval registers and index registers

Refer to [Setting Up Meter Configuration Types](#) for more information.

Designing Your Meter Types

You will need to set up appropriate [meter type codes](#) for the meters, which will be used to define or group index and interval channels. You must turn on the **Allow Interval Registers** switch for any meter type used by a meter, which will contain interval registers. Indicate the valid meter configuration types for this meter type.

NOTE:

If a meter type will only be used for meters linked to index channels, nothing special is required. Their behavior is similar to standard non-interval meters.

Designing Your Time of Use Options

Designing Your Time Of Use Rate Components

As you know from the rates chapter, the system can handle billing of quantities with different time of use periods using the SQ calculation rule type. Refer to [How To Set Up Service Quantity Calculation Rule](#) for more information. A more sophisticated rate component type is available, which can map interval profile data into time of use definitions and then apply prices based on time of use to produce multiple bill lines.

Obtain copies of existing bills that use the rate in question. If the rate is new, then write up EXACTLY how the information should appear on the customers' printed bills.

- Identify all the lines that represent charges for individual time of use periods.
- Determine how the quantities for the time of use periods are calculated. For example, what is the source data? Which time of use map is used to define the time periods? A TOU Pricing rate component type may be used to map each curve to its time of use quantities, apply time of use prices to mapped quantities or do both mapping and pricing. It all depends on how your algorithm is written.
 - In order to perform mapping, this rate component type needs to know where to get the interval data. You must indicate an Interval Profile Relationship Type (which is used to define the source data to map). Refer to [Billable Interval Quantities for a Service Agreement](#) for more information.
 - In order to perform mapping or pricing, this rate component type needs to know the classic TOU map. If the TOU map is common to ALL service agreements for the rate, you must indicate the appropriate TOU map. If the TOU map differs for different service agreements, you must indicate a TOU Map Relationship Type. The algorithms use this to find the appropriate [classic TOU map linked to the service agreement](#).
 - In order to perform pricing, this rate component type requires either a regular [bill factor](#) or a [TOU bill factor](#). The bill factor may contain the collection of prices directly, or may indicate that the prices are customer specific and can be found as contract quantities for the service agreement.
 - A calculation algorithm to map and / or price the quantities is needed. For mapping, this algorithm must apply the classic TOU map to the quantity curve and produce entries in the [calculation/audit read details](#) collection. For pricing, the algorithm will need to use the appropriate bill factor to apply prices to the quantities in the read details collection. If this bill factor is a TOU bill factor, a separate price exists for each time of use.

For our example, let's assume that curves exist for both active demand and reactive demand and assume each curve is mapped with a different classic TOU map. In addition, let's assume that excess demand used during a specific curtailment period will be priced at a different rate.

- For active demand, we have four time periods: on-peak winter, off-peak winter, on-peak summer and off-peak summer.
- For reactive demand, there are no charges in the winter so we have only two time periods: on-peak summer, off-peak summer. This classic TOU map is common to ALL service agreements for the rate.
- The curtailment charge will only appear if there is a curtailment event and the customer used more than their maximum demand defined for the period.

RC Type	UOM/TOU	Bill Factor	Interval Profile Relationship Type	Algorithm	TOU Map Relationship Type	TOU Map ID
TOU Pricing	N/a	DMDTOU	Active Demand	Map quantities to time of use periods; apply prices	Active Demand Map	
TOU Pricing	N/a	RCTVTOU	Reactive Demand	Map quantities to time of use periods; apply prices		13849374
TOU Pricing	N/a	CURTAIL	Curtailment charge	Map quantities to time of use periods, compare with maximum demand; apply prices to excess	Curtailment	

Resulting bill calculation lines from the above TOU mappings (assuming this bill crosses the summer and winter seasons and a curtailment event occurred).

Charge for active kW On peak winter nn kW @ \$0.0353/kW: \$nn.nn

Charge for active kW Off peak winter nn kW @ \$0.0298/kW: \$nn.nn

Charge for active kW On peak summer nn kW @ \$0.0483/kW: \$nn.nn

Charge for active kW Off peak summer nn kW @ \$0.0327/kW: \$nn.nn

Charge for reactive kV On peak summer nn kV @ \$0.0293/Kv: \$nn.nn

Charge for reactive kV Off peak summer nn kV @ \$0.0231/Kv: \$nn.nn

Charge for curtailment excess nn kW @ \$0.0593/kW: \$nn.nn

Once you have your rate components designed, you will be able to design the other control tables needed to set up your time of use billing customer.

NOTE:

Your time of use rate also requires interval profile relationship types. Refer to [Designing Interval Profile Relationship Types](#) and to [Designing Interval Profile Types](#).

Designing Your Time of Use Codes

The next most logical step in designing your time of use mapping controls is to define your time of use codes. To do this, look at the time of use periods to which your usage needs to be mapped. These values will likely correspond to the time of

use quantities that your rate bills for. (Although it's possible that you are not billing for every time of use period.) Be sure to consider special time periods in your contracts such as holidays, curtailment days and interruption days.

Building on the above example, we have the following time of use codes defined:

Time of Use	Description
ONWIN	On Peak Winter
OFFWIN	Off Peak Winter
ONSUM	On Peak Summer
OFFSUM	Off Peak Summer
CURTAIL	Curtailment period

For more information about time of use, refer to [UOM versus TOU versus SQI](#).

Designing Your TOU Groups

To further aid in designing time of use mapping, the TOU Group enables you to group together all the time of use codes that are used in a single map. In our above example, we will have two TOU Groups because the active and reactive energy quantities are mapped to different sets of time periods. You must also decide if you want to use a sequence number to define the relative order of a TOU within a TOU group.

FASTPATH:

Refer to [Grouping of TOU Codes](#) for more information.

TOU Group	Description	Time of Use	TOU Sequence
4PARTS	Group for a 4-part map including On and Off Peak for Winter and Summer.	ONWIN	0
		OFFWIN	0
		ONSUM	0
		OFFSUM	0
2PARTS	Group for a 2-part map including On and Off Peak for Summer	ONSUM	0
		OFFSUM	0
CURTAIL	Curtailment group, contains only one time of use	CURTAIL	0

Designing Your Classic TOU Map Relationship Types

The starting point for designing your TOU map relationship types is the rate. First, identify the relationship types that your rate needs in order to successfully map quantities.

Using the above rate as an example, only two TOU map relationship types are required because the Reactive Demand map is linked directly to the rate component. However, let's add a relationship type for reactive demand also in case it is needed in the future.

TOU Map Relationship Type	Description
---------------------------	-------------

ACTVDMD	Active Demand
RACTVDMD	Reactive Demand
CURTAIL	Curtailment

Now you must think of other types of maps that may need to be linked to the service agreement in order to successfully produce a bill. For example, are there any data derivation algorithms that require a classic TOU Map? If so, you need to define an appropriate classic TOU map relationship type to define the role for this map.

Designing Your Classic TOU Map Types

Now that you have your TOU map relationship types and your TOU groups defined, you can begin defining classic TOU map types. Recall that the classic TOU map type defines the physical attributes of the classic TOU map.

The essential attributes of any profile type are the TOU Group and the minutes per interval and the associated TOU map data creation algorithms.

The easiest way to start defining your classic TOU map types is to start with the classic TOU map relationship type and determine what type of classic TOU map each customer may have for the same TOU map relationship type.

Let's start with our TOU map relationship types and make some assumptions regarding the physical attributes of the data that a customer may have for each of these roles:

- For Active demand, the quantities may be recorded in 15-minute or 30-minute intervals. To facilitate efficiency in processing the data, we recommend that the TOU map data is stored in the same minutes per interval as the interval data being mapped. We'll assume that this is a common TOU map type.
- For Reactive demand, the quantities are recorded in 60-minute intervals. This is linked directly to the rate and is therefore a common TOU map type.
- For curtailment, we will assume that the override map will be in 15-minute, 30-minute or 60-minute intervals, based on the interval size of the actual demand data. In addition, we require a TOU map creation algorithm. This algorithm will produce a data for the curtailment TOU map based on the existence of a contract option event. It will generate the time of use data based on a [classic TOU map template](#). Refer to [Contract Option Background Topics](#) for more information. This will be an SA owned classic TOU map type.

NOTE:

We said above that we don't need a classic TOU map relationship type for Reactive Demand for our rate example. However, classic TOU map types must be defined. The table below includes the TOU map relationship type because we said we would add it just in case.

We will assume that all of our TOU data will not follow any seasonal time shifting. It will always be displayed in standard time. Refer to [Time Zone and Time Changes](#) and [Designing Your Time Options](#) for more information.

TOU Map Rel. Type	TOU Map Type, Classic	Common/ SA Owned	TOU Group	Min/ Interval	Seasonal Time Shift	Creation Algorithm
ACTVDMD	4PART15	Common	4PART	15	NoShift	N/a
	4PART30	Common	4PART	30	NoShift	N/a
RACTVDMD	2PART60	Common	2PART	60	NoShift	N/a
CURTAIL	CURT15	SA Owned	CURTAIL	15	NoShift	Create based on events for 'curtail' contract option type.

CURT30	SA Owned	CURTAIL	30	NoShift	Create based on events for 'curtail' contract option type.
CURT60	SA Owned	CURTAIL	60	NoShift	Create based on events for 'curtail' contract option type.

Now, you're ready to set up your interval billing options.

NOTE:

The above table accomplishes two steps: defining TOU map types and defining the valid classic TOU map types for a TOU map relationship type. When setting up this data, you will need to define your classic TOU map types first and then link them to the appropriate classic TOU map relationship type.

Designing Your Classic TOU Map Templates

In order to help your users to create and maintain data for classic TOU maps, you may define classic TOU map templates, which can be used to generate data for a classic TOU map. The templates may be used to define standard data for a TOU map as well as data for special periods, such as interruptions and holidays. The classic TOU map templates will reference a classic TOU map type. The system will use the TOU group and the minutes per interval from the map type to verify the setup of the template.

For our example, let us first design templates for the 15 minutes active demand TOU map type. Let's assume:

- On Peak for both winter and summer is from Monday to Friday, from 9am to 5pm, inclusive
- Off Peak for both winter and summer is all day Saturday and Sunday and from Monday through Friday, Off Peak is from 12AM to 8:45AM, inclusive and from 5:15pm to 11:45pm inclusive.

We'll worry about the season definitions for winter and summer later.

In addition, let's design the curtailment map template. Assume:

- Curtailment covers 9am to 5pm inclusive on the given curtailment day.

Designing Daily Templates

We need to define four daily templates: one for summer weekdays, one for summer weekend days, one for winter weekdays and one for winter weekend days.

Before defining the template, take note of some points about defining template components:

- For our winter weekday, we can define the off peak period with one entry covering 5:15pm through 8:45am using the Start and End Sequence Numbers to indicate that the end of the time period is on the second day.
- The classic TOU map generator will create intervals for the first period AFTER the start time up to and including the end time. As a result, our Start Time must be the interval before the first interval covered by the time of use code.

TOU Map Template, Classic	TOU Map Type, Classic	Template Type	Start Seq	Start Time	End Seq	End Time	Time of Use
15MinWntrDay	4PART15	Daily	1	8:45AM	1	5PM	ONWIN

			1	5PM	2	8:45AM	OFFWIN
15MinWntrWknd	4PART15	Daily	1	12AM	2	12AM	OFFWIN
15MinSmrDay	4PART15	Daily	1	8:45AM	1	5PM	ONSUM
			1	5PM	2	8:45AM	OFFSUM
15MinSmrWknd	4PART15	Daily	1	12AM	2	12AM	OFFSUM
15MinCurtail	CURT15	Daily	1	8:45AM	1	5PM	CURTAIL

Designing Weekly Templates

Now that we have our daily templates defined, we can define our weekly templates. For our example, we will need two weekly templates: one for a typical winter week and one for a typical summer week. For weekly templates, you must indicate the start day of the week.

We do not need a weekly template for the curtailment option.

TOU Map Template, Classic	TOU Map Type, Classic	Template Type	Week Start	Start Seq	Start Day	Start Time	End Seq	End Day	End Time	Template
15MinWntrWknd	4PART15	Weekly	Mon	1	Mon	12AM	1	Sat	12AM	15MinWntrDay
				1	Sat	12AM	2	Mon	12AM	15MinWntrWknd
15MinSmrDay	4PART15	Weekly	Mon	1	Mon	12AM	1	Sat	12AM	15MinSmrDay
				1	Sat	12AM	2	Mon	12AM	15MinSmrWknd

Finally, we need to decide if we want to create a Calendar template for our example. The decision of whether or not to create a calendar template will be based on a few factors:

- How many different seasons exist in the TOU map, where each season has different weekly templates?
- How do your holidays behave?

First, let's consider holiday behavior.

Designing Holiday Templates

When designing your classic TOU map templates, a special consideration should be made for holidays that occur throughout the year. Look at your classic TOU map contracts and ask the following questions:

- Are special prices applicable on holidays? Do these special prices require new time of use codes?
- If holidays do not get special prices, does the daily template look like the daily template for other days for this classic TOU map, for example like a weekend day?
- Do your holidays all have the same time period definitions as each other?

For our example, let's assume that our holidays use the Off Peak time of use for the season that it is in. In other words, winter holidays will use OFFWIN for every interval in that day and summer holidays will use OFFSUM for every interval in that day. As a result, we will not need to create a new template just for holidays. Instead, we can use the 15MinWntrWknd daily template for winter holidays and the 15MinSmrWknd daily template for summer holidays.

Designing Calendar Templates

The calendar template is used to define daily and weekly templates for month and day ranges. The map components may represent dates within a single year or may cross into the following year.

Remember that a main reason for creating templates is for use on the [TOU Map Generation](#) page to create data for a classic TOU map. On that page, you will specify the dates you want to generate data for. In addition you must keep in mind that you can specify one holiday template for the time period you are generating.

In our example, we have two seasons and our holiday template is different in each season. Creating a calendar template to define the seasons will not help us, because on the TOU map generation page, we wouldn't be able to indicate a single holiday template for the whole year. For our example, rather than creating a calendar template, we could simply go to the classic TOU map generation page and indicate:

- 1st Nov <year> through 30th April <year>, use the 15MinWntrWk template. For holiday overrides, use the 15MinWntrWknd daily template
- 1st May <year> through 31st October <year>, use the 15MinSmrWk template. For holiday overrides, use the 15MinSmrWknd daily template

In fact, in our example, even if our holiday template would not vary by season, one would have to weigh the advantage of creating a calendar template to define 2 seasons vs. running the TOU map generation twice. If you have a very small number of customers using the same templates, then a calendar template may not be worth creating. However, if you have many customers using the same template, it may be worth it.

Here is a possible guideline to follow for deciding whether or not to create calendar templates:

- If the same holiday template may be used for all holidays, and your customer's contracted time period definitions contain more than one season, then creating a calendar to define the seasons is recommended.
- If the holiday template changes throughout the year based on the season, then you must weigh the advantages of your possibilities:
 - You may just create weekly templates for each season and define the dates and the holiday template when using the classic TOU map generation page. This option is recommended when the templates apply to a very small number of classic TOU maps. The time taken to generate the data based on the weekly templates would be less than the time you would have spent on defining the calendar.
 - You may define your calendar to explicitly indicate your seasons and your holidays. Then when generating the data for this template, you would not indicate an override holiday template. This option is recommended when the template applies to a larger number of customers. The time taken to define the calendar would be less than that spent on generating multiple seasons for each TOU map.

Let's go ahead and design a calendar template using our weekly templates to understand the setup. For this purpose, let's assume our holiday template would be the same all year round.

Assume the seasons are defined as follows: Winter is 1st Nov. through 30th April, Summer is 1st May through 31st October. This template shows that the summer schedule includes the interval 12 AM on 01, November. This means that summer's last interval is the one that covers from 11:46 PM on the 31st of October through 12 AM on the 1st of November. The first interval in winter is 12:15 AM on the 1st of November, which covers the period from 12:01 AM through 12:15 AM. The similar setup is true for the end of winter / beginning of summer.

TOU Map Template, Classic	TOU Map Type, Classic	Template Type	Start Seq	Start Date	Start Time	End Seq	End Date	End Time	Template
15MinCindr	4PART15	Calendar	1	01May	12AM	1	01Nov	12AM	15MinSmrWk
			1	01Nov	12AM	2	01May	12AM	15MinWntrWk

Designing Your Contract Options

Designing Your Contract Option Types

Contract options are used for service agreements, which may be subject to special overrides or alterations in their rate for certain temporary periods. Contract options are used to define the possible special situations and to record the actual override events. Your contract option type is a high level categorization of the possible special situation applicable to a customer or group of customers.

To design your contract option types, you will need to review each rate for your interval customers and determine whether or not there are special options. Some examples of special options are:

- Interruptions. Perhaps you warn your customers of interruption periods and any usage incurred during that period will be subject to special prices.
- Curtailment. Perhaps you define periods where you ask a customer to shed their load below a certain subscribed demand and any demand, which exceeds this subscribed demand during that period, will be subject to a special charge.

Next, must determine whether or not a given option behaves differently under different circumstances. For example, perhaps you have different types of interruptions. If these different types of interruptions are applicable to the same customer, then you will require a single contract option type and multiple contract option event types for that option type. If these different types of interruptions are not applicable to the same customer, then you would probably want to define separate contract option type values.

You need to also consider whether or not contract option events for this type of contract option may overlap in their effective periods. If you do allow overlap, then you must be sure that any algorithm, which may process the contract option events, must know how to process the overlaps.

Your next step is to determine whether or not characteristic values will be needed for each contract option of this type. The characteristic values are available for use by the algorithms, when processing the contract options. For example, perhaps subscribed demand is defined at the contract option level. On the contract option type, you must define the possible characteristic types for contract options of this type. You may also define a default value, if applicable.

Finally, you will need to determine whether or not you wish to create validation algorithms to validate your contract option event data. Algorithms may be created to be executed upon add or change of the data, where the status is Pending, Frozen or Canceled.

For our purposes, let us assume that we have only two contract option types: Interruption and Curtailment. Neither one allows overlap. The Curtailment option type will define subscribed demand as a required characteristic and will define a validation algorithm to ensure that the value of the demand falls within an appropriate range. This algorithm program will be executed for pending and frozen events.

NOTE:

Because there are different algorithm entities, more than one algorithm type and algorithm are required, but both algorithm types may use the same program.

We will assume that both contract option types will follow the same seasonal time shift as our base time zone. Refer to [Time Zone and Time Changes](#) and [Designing Your Time Options](#) for more information.

Contract Option Type	Descr	Allow Overlap	Seasonal Time Shift	Char Type	Algorithm
INTERRUPT	Interruptions	No	USShift	N/a	N/a

CURTAIL	Curtailments	No	USShift	SubscrDmd, required, no default value	Pending: SbscrDmdPV (subscribed demand pending validation) Frozen: SbscrDmdFV (subscribed demand frozen validation) Canceled: n/a
---------	--------------	----	---------	---	--

NOTE:

Once you have the contract option types required to support your rate and derivation algorithms, we recommend that you set up Start Options for your SA Types to assist a CSR in setting up a customer for this rate. Refer to [Designing Your Start Options](#).

Designing Your Contract Option Event Types

When you designed your contract option types, you were already considering the possible contract option event types. You will need to define a different contract option event type for each different type of event that may occur for the same contract option type.

Your next step is to determine whether or not characteristic values will be needed for each contract option event of this type. The characteristic values are available for use by the algorithms, when processing the contract option events. On the contract option event type, you must define the possible characteristic types for contract option events of this type. You may also define a default value, if applicable.

For our example, we will define two types of interruptions, which may be applicable for customers on the same rate. We will define only one type of curtailment event. None of the event types require characteristic values.

Contract Option Event Type	Descr	Contract Option Type	Char Type
InterruptA	Interruptions, type A	Interrupt	N/a
InterruptB	Interruptions, type B	Interrupt	N/a
Curtail	Curtailments	Curtail	N/a

Designing Your SA Interval Billing Options

Designing Your SA Types

Your interval service agreements will require special data to be set up, such as special rates, interval profiles, TOU maps and contract options. In addition, it will need to define a cutoff time and start day option. The system requires you to indicate a special role of Interval for SA types defined for interval billing service agreements.

Refer to [Defining Service Agreement Types](#) to help you design the standard fields required for all SA Types. Additionally, for interval SA Types, you will need to define the valid Profile Relationship Types, the valid Map Relationship Types and the valid Contract Option Types for this SA Type.

Let's assume that we set up only one SA Type for all interval billing customers. As a result, all the relationship types defined in our sample are valid.

CIS Division / SA Type	Special Role	Profile Relationship Type	Map Relationship Type	Contract Option Type
CA / E-INTBIL	Interval	HEDGEVCR	ACTVDMD	INTERRUPT
		OVRHEDGE	CURTAIL	CURTAIL
		MEASDMD		
		ACTVDMD		
		RACTVDMD		

NOTE:

that this setup does not provide any link between the profile relationship types, the map relationship types and the contract option types. These are just a list of valid entries for the SA Type.

Although we created a TOU map relationship type for RACTVDMD above, it is not needed for the TOU mapping rate so it is not linked to the SA Type as a valid value.

Refer to [Setting Up SA Types](#) for more information.

Designing Your Start Options

Once you have your rates defined and your SA Type defined, you should design Start Options to aid your customer service representatives in setting up these service agreements.

You will first need to design your [start options](#) with regard to standard SA information. Note that there may be other setup required before you can add your interval billing start options:

- To link common profiles to a start option you will need to create the appropriate [profile](#) first.
- To link common classic TOU maps to a start option you will need to create the appropriate [TOU map](#) first. This is only applicable to common maps that should be linked to the service agreement. In our example, it applies to the active demand map. The reactive demand map is linked directly to the rate component so no start option information is needed.
- To link shared contract options to a start option you will need to create the appropriate [contract option](#) first.

Note that when designing our rate components above, we did not give names to their rate schedules. Let's call the interval pricing rate schedule "INTPRC" and let's call the TOU Map rate schedule "INTTOU".

Recall that the classic TOU Map rate component required profile relationship types that we did not define above. We will assume that profile relationship types are defined with the same names as the map relationship types. We will also assume that the profile types define the same interval size as the map types.

You may also use Start Options to define Cutoff Time and Start Day Option, if they are different from the values defined on the installation record. Let's assume that our TOU mapping customers use a different Cutoff Time from the installation record.

The following table shows the start option definitions with the rates and the cutoff time information.

CIS Division / SA Type	Start Option	Rate Schedule	Cutoff Time	Start Day Option
CA / E-INTBIL	Interval Pricing Hedge - 30MIN	INTPRC		
	Interval Pricing Hedge - 60 Min	INTPRC		
	TOU Map - option 1	INTTOU	2:00 AM	Current

The following table shows the profile related start options.

CIS Division / SA Type	Start Option	Rate Schedule	Profile Relationship Type	Profile Type	Profile (Profile Type)
CA / E-INTBIL	Interval Pricing Hedge - 30MIN	INTPRC	HEDGECVR		12859302 (HEDGE30)
			OVRHEDGE	OVRHG30	
			MEASDMD	DMDKW30	
	Interval Pricing Hedge - 60 Min	INTPRC	HEDGECVR		4922018 (HEDGE60)
			OVRHEDGE	OVRHG60	
			MEASDMD	DMDKW60	
TOU Map - option 1	INTTOU	ACTVDMD	ACTV30		
		RACTVDMD	RACTV30		
TOU Map - option 2	INTTOU	ACTVDMD	ACTV15		
		RACTDMD	RACTV60		

The following table shows the classic TOU map related start options.

CIS Division / SA Type	Start Option	Rate Schedule	Map Relationship Type	Map Type	TOU Map (Map Type)
CA / E-INTBIL	TOU Map - option 1; no curtailment	INTTOU	ACTVDMD		12859302 (4PART30)
	TOU Map - option 2; no curtailment		ACTVDMD		52829947 (4PART15)
CA / E-INTBIL	TOU Map - option 3; curtailment	INTTOU	ACTVDMD		12859302 (4PART15)
			CURTAL	CURT15	

The following table shows the contract option related start options.

CIS Division / SA Type	Start Option	Rate Schedule	Contract Option Type	Contract Option
CA / E-INTBIL	TOU Map - option 3; curtailment	INTTOU		48399239048 (CURT15)

Let's summarize what these tables define:

- The INTPRC rate requires 2 SA Owned profiles and one Common profile. The Common Profile needs to be created and linked to the start option. (For clarification, the Profile Type used for the profile is displayed in parentheses.) For the SA Owned profile, you need to indicate the profile type.
- There are two possible configurations for the rate INTPRC. As a result, there are two start options, each with the correct configuration defined.
- For the rate INTTOU, the SA will need profiles, TOU maps and, if they use the curtailment option, contract options.
 - The profiles will be SA owned, so only the profile relationship type and profile types are indicated.

- For the TOU maps, the active demand maps is common so the actual map is indicated, whereas the curtailment map will be SA owned so only the map type is indicated. (The reactive demand map is linked directly to the rate component.)
- The curtailment option will be shared by more than one service agreement, so the contract option itself is linked to the start option.

Refer to [Setting Up Start Options](#) to learn how to enter this information.

Setting Up Interval Billing Options

Setting Up Interval Billing Control Tables

Setting Up Profile Relationship Types

Profile Relationship Types define the role that a set of interval data will serve for a customer. Open **Admin > Interval Profile Rel Type > Add** to define your profile relationship types.

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **Interval Profile Relationship Type ID** and **Description** for the profile relationship type.

Enter the **Interval Profile Type** collection. This is a list of valid [interval profile types](#) whose collections of data can serve the role defined by the profile relationship type.

NOTE:

To aid in setup, the interval profile type is not a required field on this user interface. This will help you to define your high level interval relationship types first and then define the more detailed profile types. The valid profile types will need to be linked to the appropriate relationship types prior to creating service agreement interval profiles.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_INTV_PFRETTY](#).

Setting Up Interval Profile Types

Interval Profile Types define values common to interval profiles of the same type. Open **Admin > Interval Profile Type > Add** to define your interval profile types.

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **Interval Profile Type ID** and **Description** for the interval profile type.

Indicate whether interval profiles of this type are SA Owned or Common by entering the appropriate value in the interval profile **Sub Type**.

The **Creation Priority** will be used by the [Interval Profile Derivation Process](#) to determine the order in which the data for profiles linked to an SA should be derived. The values range from 10, being the highest priority to 90 being the lowest priority.

NOTE:

The values for this field are customizable using the Lookup table. This field name is CRE_PRIO_FLG.

Enter the **Unit of Measure** (UOM) that data values stored for interval profiles of this type are captured in.

If a service quantity identifier is needed to further qualify data stored for interval profiles of this type, enter a valid **SQ Identifier**.

Enter the **Minutes per Interval** to define the number of minutes expected in between each row of data collected for interval profiles of this type.

If the [installation](#) record indicates that [seasonal time shift](#) is required, then you must enter the appropriate **Seasonal Time Shift** record applicable for the interval data. Please take special note of the issue described in the [Evenly Sized Intervals](#) section.

The grid contains **Algorithms** that may be used to create or validate interval data for profiles of this type. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** number and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

We expect that you will need more algorithms than we supply. Your algorithms will be based on any number of factors. Be aware that new algorithms may require programming. See [How To Add A New Algorithm](#) for more information.

The following table describes each **System Event**.

System Event	Description
Interval Data Creation	These types of algorithms are used to derive interval data for a profile. Click here to see the algorithm types available for this system event.
Interval Data Validation	These types of algorithms are used to validate profile data . Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_INTV_PF_TYP](#).

Setting Up Interval Pricing Rate Components

This section describes how to define your interval pricing rate components.

NOTE:

Define Rate Schedule, Rate Version and other Rate Components. Before creating your interval pricing rate component, you must set up a rate. During this process, you will also need to set up calculation groups and rules. Refer to [Rates](#) for more information.

When you are ready to set up your interval pricing rate component, open **Menu > Rates > Rate Version Classic > Search**. Choose the rate version that this component should belong to. Use the Rate Version context menu and select Go to Rate Component + to add a new rate component.

Description of Page

Refer to [Rate Component - Main Information](#) for information about the common fields on this page. When defining an Interval Pricing rate component additional fields become available to you. The following information will help you to set up your Interval Pricing rate components.

Indicate whether or not this is **FCPO**.

Select a **Value Type** of Unit Rate. This field will be gray when the rate component is referenced on another rate component. The **Value Source** will most likely be Billing Factor. Your bill factor will likely be one with a type of Interval.

FASTPATH:

Refer to [Defining Interval Values](#) for more information about setting up bill factors with interval prices.

Indicate whether or not this rate component is **Seasonal**. Refer to [Rate Component - Main Information](#) for more information about seasonal rate components.

The **Error if No Value** field is available for you to use in the calculation algorithm.

Indicate the **Calc Algorithm** that the system will use to calculate the bill line that this rate component produces.

- The system provides an Algorithm Type that is available for use here. It is called **RCIPRS**.

If this algorithm does not provide you with the logic you require, you will need to create a new algorithm (refer to [Setting Up Algorithms](#)). The above existing algorithm should be used as a sample if you have to write a new algorithm type.

NOTE:

The calculation algorithm's main purpose is to create bill calculation lines. However, the algorithm may populate other information for the bill, for example, it may add to the SQ or register read collection or it may overwrite the description on bill.

Indicate the **Audit Algorithm** to be used when a CSR wants to drill down into the details of a bill line that was calculated using this Rate Component.

- The system provides an algorithm type that is available for use here. It is called **RCIPRS-ADT**.

If this algorithm does not provide you with the logic you require, you will need to create a new algorithm (refer to [Setting Up Algorithms](#)). The above existing algorithm should be used as a sample if you have to write a new algorithm type.

NOTE:

The audit algorithm should produce the same results as the Calc Algorithm. They are separated because they have different responsibilities. For example, the Calc Algorithm should produce bill lines, but the Audit Algorithm should not. They share common logic related to accessing and processing the appropriate interval data records. As a result, it is recommended that these two programs share a common code which accesses and processes the interval data. The above algorithms provided by the system behave this way and should be used as samples.

Refer to [Interval Billing Calculation Details](#) to understand where a CSR may view the calculation details for a bill calc line, using this algorithm.

Indicate the **Interval Profile Rel Type**. This indicates to the system the profile, linked to the SA, which contains the interval quantities to be processed. Refer to [Business Role of Interval Data](#) and [Setting Up Profile Relationship Types](#) for more information.

Turn on **GL Statistical Quantity** if GL journal lines generated for this rate component should also contain the service quantity amount as a statistical quantity. You would use this option if you keep track of both dollar amounts and consumption units in your general ledger.

Enter the verbiage to appear on the customer's bill in **Description On Bill** and turn on the **Print** switch. Refer to [Rate Version - Bill Print Info](#) for more information about these fields.

Move to the [Rate Component - GL Distribution](#) window to define how to book moneys associated with this rate component in the general ledger.

Setting Up Channel Control Tables

Setting Up Interval Register Types

FASTPATH:

Refer to [The Big Picture of Raw Data Collection and Aggregation](#) for more information.

Interval register types define values common to interval registers of the same type. Open **Admin** > **Interval Register Type** > **Add** to define your interval register types.

NOTE:

This page will not be available if Meter Data Management module is [turned off](#).

Description of Page

Enter a unique **Interval Register Type** and **Description** for the interval register type.

Enter the **Unit of Measure** (UOM) that data values stored for interval registers of this type are captured in.

If a service quantity identifier is needed to further qualify data stored for interval registers of this type, enter a valid **SQ Identifier**.

Enter the **Minutes per Interval** to define the number of minutes expected in between each row of data collected for interval registers of this type.

If the [installation](#) record indicates that [seasonal time shift](#) is required, then you must enter the appropriate **Seasonal Time Shift** record applicable for the interval data. Please take special note of the issue described in the [Evenly Sized Intervals](#) section.

The grid contains **Algorithms** that may be used to perform actions on interval data for registers of this type. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** number and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

We expect that you will need more algorithms than we supply. Your algorithms will be based on any number of factors. Be aware that new algorithms may require programming. See [How To Add A New Algorithm](#) for more information.

The following table describes each **System Event**.

System Event	Description
Interval Register Validation	These types of algorithms are used to validate register data . Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_INTV_REG_TYP](#).

Setting Up Meter Configuration Types for Channels

You must set up [meter configuration types](#) for your interval channels and index channels.

Setting Up Meter Types

You must set up [meter types](#) to use for grouping together your index and interval registers. You must turn on the **Allow Interval Registers** switch for any meter type used by a meter, which will contain interval registers. Indicate the valid meter configuration types for this meter type.

Setting Up Time of Use Billing Control Tables

Setting Up Classic TOU Map Relationship Types

Classic TOU Map Relationship Types define the role that a classic TOU Map will serve for a customer. Open **Admin > Consumption > TOU Map Relationship Type > Add** to define your classic TOU map relationship types.

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **TOU Map Relationship Type ID** and **Description** for the TOU map relationship type.

Enter the **TOU Map Type** collection. This is a list of valid [TOU map types](#) whose collections of data can serve the role defined by the TOU map relationship type.

NOTE:

To aid in setup, the map type is not a required field on this user interface. This will help you to define your high level map relationship types first and then define the more detailed map types. The valid map types will need to be linked to the appropriate relationship types prior to creating service agreement TOU maps.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TMAP_RELTY](#).

Setting Up TOU Pricing Rate Components

This section describes how to define your TOU Pricing rate components. Refer to [Time of Use Mapping and Pricing](#) for background information.

NOTE:

Define Rate Schedule, Rate Version and other Rate Components. Before creating your TOU pricing rate component, you must set up a Rate Schedule. Refer to [Setting Up A Rate Schedule](#) for more information. You will also need to set up a rate version and possibly set up other types of rate components. Refer to [Defining Rate Versions](#) and [Defining Rate Components](#) for more information.

When you are ready to set up your TOU pricing rate component, open **Menu > Rates > Rate Version Classic > Search**. Choose the rate schedule and version that this component should belong to. Open its context menu, and select **Go to Rate Component Classic > Add** to add a new rate component.

Description of Page

Refer to [Rate Component - Main Information](#) for information about the common fields on this page. When defining a TOU Pricing rate component, additional fields become available to you. The following information will help you to set up your TOU Pricing rate components.

If the algorithm used by this rate component, only performs mapping and does not produce bill lines, indicate that this is **FCPO**.

The **Value Type** and **Value Source** are optional. If your algorithm performs pricing logic, then you will probably need to define a bill factor here. If your calculation algorithm only performs mapping, then you may not require any value here.

Indicate whether or not this rate component is **Seasonal**. Refer to [Rate Component - Main Information](#) for more information about seasonal rate components.

The **UOM** is available for use by your calculation algorithm. It could be used, for example, by a mapping algorithm to produce SQ quantities with a different unit of measure, assuming that the algorithm knows how to convert from one UOM to another. Note that the algorithms provided with the system do not use this field.

The **SQI** is available for use by your calculation algorithm. It could be used, for example, by a mapping algorithm to produce SQ quantities with a different SQI. Note that the [TOU mapping and pricing](#) algorithms provided with the system will use this field, if populated, to produce SQI quantities with this SQI value.

Measures Peak Qty and **GL Statistical Qty** behave the same way they do for SQ quantity rate components. Refer to [How To Set Up Service Quantity Rate Components](#) for more information.

The **Error if No Value** field is available for you to use in the calculation algorithm.

Indicate the **Calc Algorithm** that the system will use to map the interval quantities to time of use values.

- The system provides the following classic TOU mapping/pricing algorithms that are available for use here: **RCTMS** performs mapping of time of use only and **RCTMPS** performs both mapping and pricing, producing the necessary bill calculation lines.

If these algorithms do not provide you with the logic you require, you will need to create a new algorithm (refer to [Setting Up Algorithms](#)). The above existing algorithms should be used as a sample if you have to write a new algorithm type.

NOTE:

The calculation algorithm's main purpose is to create bill calculation lines. However, the algorithm may populate other information for the bill, for example, it may add to the SQ or register read collection or it may overwrite the description on bill.

Indicate the **Audit Algorithm** to be used when a CSR wants to drill down into the details of a bill line that was calculated using this Rate Component.

- The system provides the following audit algorithm that is available for use here: [RCTPRS-ADT](#).

If this algorithm does not provide you with the logic you require, you will need to create a new algorithm (refer to [Setting Up Algorithms](#)). The above existing algorithm should be used as a sample if you have to write a new algorithm type.

NOTE:

The audit algorithm should produce the same results as the Calc Algorithm. They are separated because they have different responsibilities. For example, the Calc Algorithm should produce bill lines, but the Audit Algorithm should not. They share common logic related to accessing and processing the appropriate interval data records. As a result, it is recommended that these two programs share a common code which accesses and processes the interval data. The above algorithms, provided by the system, behave this way and should be used as samples.

Refer to [Interval Billing Calculation Details](#) to understand where a CSR may view the calculation details for a bill calc line, using this algorithm.

Indicate the **Interval Profile Rel Type**. This indicates to the system the profile, linked to the SA, which contains the interval quantities to be mapped. Refer to [Business Role of Interval Data](#) and [Setting Up Profile Relationship Types](#) for more information.

If the classic TOU map differs for different service agreements, indicate the **TOU Map Relationship Type**. The system uses this information to find the correct classic TOU map linked to the SA. Refer to [Business Role of a TOU Map](#) and [Setting Up TOU Map Relationship Types](#) for more information.

If ALL service agreements linked to the rate use the same map, indicate the classic **TOU Map ID**.

Enter the verbiage to appear on the customer's bill in **Description On Bill** and turn on the **Print** switch. Refer to [Rate Version - Bill Print Info](#) for more information about these fields.

Setting Up Time of Use Codes

An important step in preparing for Time of Use billing is to define Time of Use codes. Refer to [Setting Up Time-Of-Use Codes](#) for more information.

Setting Up TOU Groups

Once you have your time of use codes defined, you will need to create your TOU groups to group the codes together. Refer to [Setting Up TOU Groups](#) for more information.

Setting Up Classic TOU Map Types

The Classic TOU Map Type defines characteristics that are common to classic TOU Maps of the same type. Open **Admin > TOU Map Type Classic > Add** to define your TOU map types.

NOTE: The Classic TOU Map Type portal describes TOU Map Types that are used with the classic rate engine. For information about the differences between the classic and new style rate engines, see [Rates](#). For details on setting up TOU Map Types for the new style rate engine, see [Creating New Style TOU Map Types](#).

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **TOU Map Type ID** and **Description** for the classic TOU Map Type.

Indicate whether TOU maps of this type are SA Owned or Common by entering the appropriate value in the **Sub Type**.

The **Creation Priority** will be used by the [TOU map data creation](#) process to determine the order in which the data for classic TOU maps linked to an SA should be derived. The values range from 10, being the highest priority to 90 being the lowest priority.

NOTE:

The values for this field are customizable using the Lookup table. This field name is CRE_PRIO_FLG.

Enter the **TOU Group** that defines the collection of TOUs for this map type.

Enter the **Minutes Per Interval** to define the number of minutes expected in between each row of data collected for classic TOU maps of this type.

If the [installation](#) record indicates that [seasonal time shift](#) is required, then you must enter the appropriate **Seasonal Time Shift** record applicable for the interval data. Please take special note of the issue described in the [Evenly Sized Intervals](#) section.

The grid contains **Algorithms** that may be used to create classic TOU map data for TOU maps of this type. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** number and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

We expect that you will need more algorithms than we supply. Your algorithms will be based on any number of factors. Be aware that new algorithms may require programming. See [How To Add A New Algorithm](#) for more information.

The following table describes each **System Event**.

System Event	Description
TOU Map Creation	These types of algorithms are used to create TOU map data automatically. Click here to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_TMAP_TYPE](#).

Setting Up Classic TOU Map Templates

The TOU Map Template classic page enables you to define daily, weekly and calendar templates for use by TOU data generation and automatic TOU data creation. Open **Admin > TOU Map Template Classic > Add** to define your TOU map templates.

NOTE:

The classic TOU Map Template portal describes TOU Map Templates that are used with the classic rate engine. For information about the differences between the classic and new style rate engines, see [Rates](#). For details on setting up TOU Map Templates for the new style rate engine, see [Creating New Style TOU Map Templates](#).

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **TOU Map Template** name and a **Description** for the classic TOU map template.

Enter a classic **TOU Map Type**. Daily templates use the TOU map type to determine the TOU group whose TOU codes are valid for this template. All templates use the classic TOU map type to determine the interval size to ensure that start and end times are valid for the interval size. For example, if your interval size is 60 minutes, then the amount of minutes between your start and end times must be a multiple of 60. For example, you would not be able to enter a start time of 10:00 and an end time of 10:15.

Use the **TOU Map Template Type** to indicate whether this template is Daily, Weekly, or Calendar.

The fields in the remaining portion of the page will depend on the classic TOU map template type.

A Daily template is used to define time of use periods applicable for hour/minute time ranges in an abstract day. Data in this template has no association with a specific day of the week or a specific calendar date. Each entry in the collection is used to define the time periods applicable for a given Time of Use code in this day. Enter the following information for your collection of time periods for the daily template.

Use the **Start Sequence Number** and the **End Sequence Number** to indicate whether the time period covered by this entry starts and ends on the same day or on different days. If the time period covered by the entry starts and ends on the same day, then the same number should be entered in both fields. If the time period ends on a different day (most likely one day later) than it starts, then the End sequence would be greater than the start sequence.

Enter the **Start Time** and **End Time** applicable for this **Time of Use** code. The valid time of use codes are limited to those belonging to the TOU group on the template's TOU map type.

NOTE:

The times on the TOU map components are in legal time. During the generation process the system will convert the time definitions into standard time, taking daylight savings into account. Time shifting is based upon the Seasonal Time Shift defined on the template's classic TOU Map Type. Refer to [Seasonal Time Shift](#) for more information.

When the TOU data is generated, intervals for this time of use code will be generated starting from the first time period AFTER the Start Time up to and including the End Time interval. Therefore, the end Time is inclusive and the start time is not inclusive.

A Weekly template is used to define the collection of daily templates that make up an abstract week. Data in this template has no association with specific calendar dates. You must indicate the **Week Start Day** to tell the system which day of the week is considered "day one".

Enter the following information for your collection of daily templates for the weekly template.

Use the **Start Sequence Number** and the **End Sequence Number** to indicate whether the time period covered by this entry starts and ends in the same week or in different weeks. If the days covered by the entry are in the same week, then the same number should be entered in both fields. If the time period ends in a different week (most likely one week later) than it starts, then the End sequence would be greater than the start sequence.

Enter the **Start Week Day** and **Start Time** and **End Week Day** and **End Time** applicable for this **Reference TOU Map Template**. The valid templates are limited to those that are daily and those whose TOU map type has the same TOU group and minutes per interval as this template's TOU map type.

NOTE:

The times on the TOU map components are in legal time. During the generation process the system will convert the time definitions into standard time, taking daylight savings into account. Time shifting is based upon the Seasonal Time Shift defined on the template's TOU Map Type. Refer to [Seasonal Time Shift](#) for more information.

When the TOU data is generated, intervals will be generated starting from the first time period AFTER the Start Time up to and including the End Time interval. Therefore, the end Time is inclusive and the start time is not inclusive.

A Calendar template is used to define the collection of weekly and daily templates that make up specific months and dates for a given calendar.

Enter the following information for your collection of daily and weekly templates for the calendar template.

Use the **Start Sequence Number** and the **End Sequence Number** to indicate whether the time period covered by this entry starts and ends in the same year or in different years. If the dates covered by the entry are in the same year, then the same number should be entered in both fields. If the time period ends in a different year (most likely one year later) than it starts, then the End sequence would be greater than the start sequence.

Use the **Start Date** (month and day) and **Start Time** and the **End Date** (month and day) and **End Time** to indicate the time period applicable for this **Reference TOU Map Template**. The valid templates are limited to those that are daily and weekly and those whose TOU map type has the same TOU group and minutes per interval as this template's TOU map type.

NOTE:

The times on the TOU map components are in legal time. During the generation process the system will convert the time definitions into standard time, taking daylight savings into account. Time shifting is based upon the Seasonal Time Shift defined on the template's TOU Map Type. Refer to [Seasonal Time Shift](#) for more information.

When the TOU data is generated, intervals will be generated starting from the first time period AFTER the Start Time up to and including the End Time interval. Therefore, the end Time is inclusive and the start time is not inclusive.

Where Used

The [generate](#) button on the TOU map page allows you to specify a TOU map template and a holiday TOU map template to use for generating TOU data.

If you have a TOU map creation algorithm defined on a TOU map type, this algorithm will need to use a TOU map template to create TOU data. The mechanism for defining the TOU map template to use depends on how the algorithm is designed. The TOU map creation algorithm provided by the system [ITMCCOPT](#) expects the template to be defined as a characteristic of the service agreement.

Setting Up Contract Option Control Tables

This section describes the pages related to maintaining contract options.

Setting Up Contract Option Types

This section describes the pages related to maintaining a contract option type.

Contract Option Type - Main

Contract Option Type defines control information required for contract options. Open **Admin > Consumption > Contract Option Type > Add** to define your contract option types.

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **Contract Option Type ID** and **Description**.

Indicate whether you **Allow Overlap** of the effective period for contract events linked to contract options of this type.

If the [installation](#) record indicates that [Seasonal Time Shift](#) is required, then you must enter the appropriate **Seasonal Time Shift** record applicable for the contract option event data.

Select a **Characteristic Type** that may be used on contract options of this type. The characteristic type's Description appears adjacent.

The following fields should be defined for each characteristic type:

Sequence This field controls the order in which the characteristics appear on the [contract option](#) page.

Required Turn this switch on if this type of characteristic must be defined on all contract options of this type.

Default Turn this switch on if this characteristic type should automatically appear in the characteristic scroll area of contract options of this type.

Characteristic Value If a characteristic value can default for contract options of this type, specify the default value in this field.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COP_TYPE](#).

Contract Option Type - Algorithms

To define validation algorithms for contract option events linked to contract options of this type. Open **Admin > Consumption > Contract Option Type > Search** and navigate to the **Algorithms** page to define validation algorithms for this contract option type.

Description of Page

The grid contains **Algorithms** that may be used to validate contract option event information. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** number and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Description
Contract Option Event Cancel	This algorithm is executed when a contract option event for a contract option of this type is canceled. Click here to see the algorithm types available for this system event.
Contract Option Event Freeze	This algorithm is executed when a contract option event for a contract option of this type is frozen. Click here to see the algorithm types available for this system event.
Contract Option Event Pending	This algorithm is executed when a contract option event for a contract option of this type is added or changed and the status is pending.

Setting Up Contract Option Event Types

Contract Option Event Type defines control information required for contract option events. Open **Admin > Contract Option Event Type > Add** to define your contract option event types.

NOTE:

This page will not be available if Complex Billing module is [turned off](#).

Description of Page

Enter a unique **Contract Option Event Type** code and **Description**.

Indicate the **Contract Option Type** for this event type.

Use the **Characteristics** collection to define characteristics that can be defined for contract option events of this type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on contract option events of this type. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_COP_EVT_TYPE](#).

Defining Prepaid Metering Options

The topics in this section describe how to set up the system to enable prepaid metering functionality.

NOTE:

Prepaid metering is optional. The system configuration requirements described in this section are only relevant if your organization offers prepaid metering service.

The Big Picture of Prepaid Metering

Prepaid metering allows customers to pay for energy before it is actually used.

How Does Prepaid Metering Work?

A traditional electronic prepaid metering system typically operates at three levels:

- Meters that are installed at the customer's home. These meters dispense energy up to the amount that the customer prepaid.
- Vending stations located at the utility's offices or designated payment agencies. These stations/agencies sell prepaid credits to customers.

- Master stations that group vending stations together for administration, reporting and control purposes. Master stations communicate information - e.g. customer information, tariff changes, etc - to the vending stations. The vending stations, in turn, report detailed customer sales and other transaction information to the master station.

A prepaid customer goes to a vendor and purchases credits for his/her meter to dispense energy. The vendor has knowledge of the customer and the rate that the customer is paying. The vendor issues the credits in any of the following forms: a prepaid card, a token (to be inserted into the meter) or a series of numbers (to be keyed into the meter).

Debit Meter vs. Credit Meter

Prepaid meters are commonly referred to as debit meters. Billed meter types are also called credit meters. A meter's state of being 'credit' or 'debit' translates to a specific meter configuration. Having this property set at the meter configuration level enables meters to be switched between 'credit' and 'debit' if needed.

Prepaid credits may be purchased for either new premises or existing premises. In the latter case, the existing premise could initially have a typical 'credit' meter. In this case, fieldwork may be involved in getting the meter switched from 'credit' to 'debit' (prepaid).

Prepaid Metering Transactions Result in Adjustments

There could be numerous types of prepaid transactions. But the most common types are prepaid sales and cancellations.

A sale or cancellation transaction includes a total amount that usually includes tax. In order to post to the general ledger, the prepaid transaction must result in a financial transaction. Moreover, separate GL entries must be created for the breakdown of revenue and tax. In some cases, it may also be necessary to calculate the usage from the revenue amount.

[Calculated Adjustments](#) are used to book revenue from these types of transactions. A base Generate Adjustment algorithm is available to let rate application generate the appropriate GL calculation lines and calculate the corresponding usage, if needed.

Prepaid adjustments do not affect a customer's balance.

Interfacing Prepaid Transactions

An adjustment upload batch process exists for uploading prepaid transactions that result in adjustments. Refer to [Interfacing Adjustments From External Sources](#) for more information.

Prepaid metering options need only be set up if your organization offers prepaid metering service to your customers. Refer to [Defining Prepaid Metering Options](#) for more information.

Prepaid Transactions Can Go Into Suspense

A prepaid transaction goes into suspense if a prepaid SA could not be determined for any of the following reasons:

- The badge number on the transaction is not a valid meter in the system
- The badge number on the transaction is a valid meter in the system, but is:
 - Not linked to a prepaid meter configuration type
 - Not linked to a non-closed, non-canceled prepaid SA

This can be a common occurrence because of the likelihood that transactions get uploaded prior to meter installation/exchange or service agreement information being updated in the system.

When this situation happens, you probably still want to recognize the revenue by posting an adjustment to a suspense SA that you designate. Refer to the base sample Get Prepaid SA Using Badge Number (C1-SABYBADGE) algorithm for an example of how the suspense SA is specified.

A batch process exists for automatically resolving suspense adjustments. Refer to [How Are Suspense Adjustments Resolved](#) for more information.

Setting Up The System To Enable Prepaid Metering

The following sections describe the steps in setting up control information for prepaid metering.

Meter Types

Prepaid meters are set up just like any other meter - i.e. the meter must reference a meter type and the meter's configuration must reference a meter configuration type.

To set up prepaid meter types you must do the following:

- Define the meter configuration types that will be used for recording prepaid usage. Mark each of these configuration types as Prepaid . Define one register with the appropriate UOM (e.g. kWh). See [Setting Up Meter Configuration Types](#) for more information.
- Define the meter types that will be used for prepaid metering. For each meter type, associate one of the Prepaid meter configuration types that you defined. You can also mark each meter type as Prepaid Capable. See [Setting Up Meter Types](#) for more information.

NOTE:

Base logic does not require the meter type to be marked as Prepaid Capable. The Prepaid indicator on the meter configuration type is used to identify prepaid meters.

Characteristic Types

The following characteristic types are needed if you are going to upload prepaid transactions as adjustments and need the system to determine the prepaid SA given a prepaid meter's badge number.

Refer to [Setting Up Characteristic Types](#) for more information.

Prepaid Entity Characteristic Type

Create a characteristic type that identifies whether an SA Type is used for prepaid metering. For example:

- Characteristic Type = <code>
- Description = Prepaid Entity
- Char Entities = SA Type
- Subclass = predefined list
- Values = Y, N

Badge Number Characteristic Type

Create a characteristic type for specifying a badge number. For example:

- <code>
- Description = Badge Number
- Char Entities = Adjustment, Adjustment Type
- Subclass = Adhoc

Algorithms

The following algorithms need to be set up in order to store prepaid transactions in the system.

Refer to [Setting Up Algorithms](#) for more information.

NOTE:

The following sections describe basic set up needed for uploading 'sale' and 'cancellation' types of prepaid transactions. Your implementation team may have to define additional specific algorithms types for any other prepaid transaction type that you need to upload into or store in the system.

Calculation Rule - Calculation Algorithm

Create an algorithm of type Determine Percent Given Total (C1-PCTGVNTOT) specifying your parameter values for UOM / TOU / SQL.

Create an algorithm of type Back Into Revenue (C1-BACKINREV) specifying your parameter values for UOM / TOU / SQL.

Adjustment Type - Generate Adjustment

Create an algorithm of type Adjustment Generation - Apply Rate (ADJG-RT), specifying your [prepaid rate](#) and parameter values for UOM / TOU / SQL.

Adjustment Type - Determine SA

Create an algorithm of type Get Prepaid SA Using Badge Number (C1-SABYBADGE), specifying your badge number characteristic type, prepaid characteristic type and suspense SA ID (i.e. SA to which the adjustment needs to post when a valid prepaid SA is not found).

Adjustment Type - Resolve Suspense

Create an algorithm of type Cancel Suspense Adjustment (C1-CANSUSADJ) specifying the following:

- [Badge number characteristic type](#)

- [Prepaid entity characteristic type](#)
- [Prepaid adjustment type](#)
- Adjustment cancel reason

If you want this algorithm to create to do entries for adjustments that have been in suspense for too long, specify additional parameter values for number of days and to do type and to do role (optional).

Adjustment Type - Adjustment Information

Specify the SUSPENSE_DESCR parameter on your adjustment information algorithm if you want the information to include an indication of suspense.

Installation - Adjustment Information

Specify the SUSPENSE_DESCR parameter on your adjustment information algorithm if you want the information to include an indication of suspense.

Rate

In order to generate separate calculation lines for the revenue and tax breakdown of the prepaid amount, you need to set up a prepaid rate with components that calculate revenue amount and tax amount given just a total amount.

Create a rate schedule with the following components:

- Tax Calculation Rule
 - Calculation Rule Type = Calculation Algorithm
 - Value Type = Percentage
 - Value Source = Bill Factor
 - Bill Factor = your tax bill factor
 - Calculation Algorithm = the one created for algorithm type C1-PCTGVNTOT. Refer to [Calculation Rule - Calculation Algorithm](#) for information.
 - Distribution Code = *your distribution code for tax liability*
- Revenue Calculation Rule
 - Calculation Rule Type = Calculation Algorithm
 - Derive SQ = checked
 - Value Type = Unit Charge
 - Value Source = Bill Factor
 - Bill Factor = your rate/kWh bill factor
 - UOM/TOD/SQI to use for the resulting calculated usage, for example kWh
 - RC Cross Reference = indicate the sequence for the tax calculation rule (above)
 - Calculation Algorithm = the one created for algorithm type C1-BACKINREV. Refer to [Calculation Rule - Calculation Algorithm](#) for information.
 - Distribution Code = your distribution code for revenue

Adjustment Types

You must define an adjustment type for each type of transaction that you want to store in the system. Refer to [Setting Up Adjustment Types](#) for more information.

For prepaid sales and cancellation transactions, you need to define [calculated adjustment types](#). In addition, plug-in the following algorithms:

- Adjustment FT Creation - GL Only algorithm - so that transaction amounts do not affect the customer's balance. Specify calc lines as the source for adjustment distribution codes.
- [Generate Adjustment](#)
- [Determine SA](#)
- [Resolve Suspense](#), if applicable.
- [Adjustment Information](#), if applicable

Adjustment Type Profiles

Create appropriate [adjustment type profiles](#) for your prepaid service agreements.

Service Agreement Types

You must create a prepaid [SA Type](#) for each unique CIS Division. Each SA Type must be set up as non-billable and requiring a characteristic premise. The SA Types must also indicate that they are used for prepaid by specifying a [prepaid entity characteristic](#) value.

Bill Cycle

Create at least one bill cycle that will be used for prepaid accounts. This bill cycle should not have a schedule defined.

Vendor Information

A vendor could be a vending station, payment agency or master station.

You must create a [person](#) for each of your vendors. Information about the vendor may include names, an address, phone numbers, etc.

Any other miscellaneous information about the vendor can be stored as characteristics on the person.

Hierarchical relationships between vendors can be established through characteristics. For instance, a vending station's person record may contain a foreign key characteristic that points to the master station that the vending station reports to.

Conservation Programs

Oracle Utilities Customer Care and Billing allows you to define conservation (or energy efficiency) programs and provide rebates to customers.

The Big Picture of Conservation Programs

The purpose of using conservation programs is to provide rebates to customers based on eligibility and verification of newly purchased appliances and hardware that are rated to conserve the demand for energy. To redeem their rebates, customers have to submit a rebate application to the utility with receipts, and the utility has to administer and report on the programs.

Admin and transaction objects are provided in the product to support the definition of conservation programs (admin data) and the subsequent rebate claims (transaction data). Portals and BPA scripts are used to maintain the conservation programs and rebate claims. Adjustments are used to recognize the expense and to process refunds.

The following sections discuss the maintenance objects that support this functionality.

Conservation Program Maintenance Object

A conservation program is an admin maintenance object (MO) used to support the definition of a conservation program. It holds rules that control how the claims for a conservation program are managed. If your organization wishes to use this MO, you can either use the business object (BO) supplied in the base product or configure your own BO.

This MO provides the following functionality:

- A business object option Display Statistics Service Script is provided for business objects of this type. The script plugged into this option retrieves information displayed on the **Conservation Program Statistics Zone**.
- A business object option Display Statistics UI Map is provided for business objects of this type. This is the map used on the **Conservation Program Statistics Zone** to display statistics.
- A separate maintenance object is provided to capture rebate definitions for a conservation program
- Logs are not provided.
- The standard characteristics collection is provided.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CONSV_PROG](#).

FASTPATH:

For more information, about this MO and to review the business objects defined for this MO, navigate to **Admin > Database > Maintenance Object > Search** and view the MO C1-CPROG.

Conservation Program Rebate Definition Maintenance Object

A conservation program has rebate definitions that define the rebate amounts and energy savings for different types of products. The superset of such lines is the "rebate matrix". Each row in the matrix defines the rebate and presumed energy savings for a product purchased by the consumer. If your organization wishes to use this maintenance object (MO), you can either use the business object (BO) supplied in the base product or configure your own BO.

This MO provides the following functionality:

- The Rebate Definition table can be used to capture information for rebate claim lines such as, appliance type, refund amount, energy savings, and applicable manufacturers and models.
- Logs are not provided for the Rebate Definition.

- A characteristics collection is not provided.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_REBATE_DEFN](#).

FASTPATH:

For more information, about this MO and to review the business objects defined for this MO, navigate to **Admin > Database > Maintenance Object > Search** and view the MO C1-RDEF.

Rebate Claim Maintenance Object

When a customer files a claim for a rebate, a rebate claim will be created. The rebate claim maintenance object (MO) is used to support the definition of a claim. If your organization wishes to use this MO, you can either use the business object (BO) supplied in the base product or configure your own BO.

This MO provides the following functionality:

- Depending on the utility's requirements, you can specify single or multiple items on the Rebate Claim. A separate maintenance object is provided to capture rebate claim lines for a rebate claim.
- The standard characteristics collection is provided.
- A log is provided. This can be used to track approval information for the claim, to capture adjustments created as the claim is processed, etc.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_REBATE_CLAIM](#).

FASTPATH:

For more information, about this MO and to review the business objects defined for this MO, navigate to **Admin > Database > Maintenance Object > Search** and view the MO C1-RCLAIM.

Rebate Claim Line Maintenance Object

A rebate claim has a rebate claim line for each product eligible for a refund. You use the rebate claim line maintenance object (MO) to create a rebate line. If your organization wishes to use this MO, you can either use the business object (BO) supplied in the base product or configure your own BO.

This MO provides the following functionality:

- SA (service): The SA table is the real service SA. There is a separate SA under which the rebate adjustments are stored.
- Logs are not provided for Rebate Claim Line
- The standard characteristics collection is provided.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_REBATE_LINE](#).

FASTPATH:

For more information, about this MO and to review the business objects defined for this MO, navigate to **Admin > Database > Maintenance Object > Search** and view the MO C1-REBLN.

GL Accounting Example

The following table shows the financial transactions that are issued when processing energy conservation rebates. This example shows the financial transactions when a customer files a claim for a dishwasher and insulation.

Notice how the expense adjustments are atomized:

- Separate adjustments are created to expense each item individually.
- A single adjustment is created for the A/P check request.

Event	GL Accounting
Rebate adjustment is created - dishwasher	Rebate Expense 35 A/P <35>
Rebate adjustment is created - insulation	Rebate Expense 45.45 A/P <45.45>
A/P check request issued	A/P 80.45 Cash <80.45>

Setting Up Conservation Programs

Conservation Programs allow administrators to create and maintain conservation programs for which customer can submit rebate claims. To set up a conservation program, open **Admin > Customer > Conservation Program > Add**.

FASTPATH:

For additional information on working with rebate claims, see [Rebate Claims](#).

The topics in this section describe the base-package zones that appear on the Conservation Program portal.

Conservation Program List Zone

The Conservation Program [List zone](#) lists every conservation program. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent conservation program.
- Click the Add link in the zone's title bar to add a new conservation program.

This zone displays the following for each conservation program defined in the system:

- Conservation Program Information (Description, Status, Start Date, End Date)
- The number of Complete Claims
- The total Complete Claims Amount
- The number of Incomplete Claims
- The Statistics Date/Time is the date that the statistics were last calculated

Use the following procedure to create a new conservation program:

- Select **Admin > Customer > Conservation Program > Add** or click **Add** on the title bar of the Conservation Program [List zone](#).
- Enter basic information about the conservation program in the **Main** section, including:
 - Conservation Program code
 - Description
 - Start Date/End Date
 - Statistic Update Frequency (in hours)
- Enter **Financial Information** about the conservation program, including:
 - CIS Division
 - SA Type
 - A/P Adjustment Type
 - A/P Taxable Adjustment Type
- Enter **Approval Information** about the conservation program, including:
 - Approval To Do Type
 - Duplicate Claim To Do Role
 - Approval Levels and Threshold Amounts
- Click **Save**. To return to the Conservation Program portal without saving the new program, click **Cancel**.

To view a specific conservation program, click the broadcast icon for the conservation program you wish to view. The remaining zones in the Conservation Program portal open displaying details about the selected conservation program.

Conservation Program Zone

You use the Conservation Program zone to view and maintain individual conservation program. The Conservation Program zone displays the following information about the selected conservation program:

- Basic information about the conservation program, including
 - Conservation Program
 - Description
 - Start Date/End Date
 - Statistic Update Frequency (in hours)
 - Status
- Financial Information about the conservation program, including
 - CIS Division
 - SA Type
 - A/P Adjustment Type
 - A/P Taxable Adjustment Type
- Approval Information about the conservation program, including
 - Approval To Do Type
 - Duplicate Claim To Do Role
 - Approval Levels and Threshold Amounts

Please see the zone's help text for information about this zone's fields.

Conservation Program Status

The Status of a conservation program indicates the current state of the program within the system. Valid statuses include:

- **Pending** indicates the program is pending. This is the initial state of a conservation program when first created.
- **Active** indicates the conservation program is currently active.
- **Inactive** indicates the conservation program is currently inactive.

Conservation Program Actions

You can perform a number of actions on a conservation program, including:

- **Edit:** Used to edit a conservation program
- **Delete:** Used to delete a conservation program
- **Activate:** Used to activate a pending or inactive conservation program
- **Refresh Statistics:** Used to refresh the statistics of an active conservation program
- **Deactivate:** Used to deactivate a pending or active conservation program
- **Pend:** Used to change the status of an active or inactive conservation program to Pending

The actions available are based on the current status of the conservation program. The table below summarizes the actions available at each status.

Status	Valid Action
Pending	Edit, Delete, Activate, Deactivate
Active	Edit, Refresh Statistics, Deactivate, Pend
Inactive	Activate, Pend

Editing Conservation Programs

Use the following procedure to edit an active or pending conservation program:

- Click **Edit**.
- Edit the details of the conservation program as needed.
- Click **Save**.

Deleting Conservation Programs

Use the following procedure to delete a conservation program:

- Click **Pend** to change the status of the conservation program to Pending (if needed).
- Click **Delete**.
- Click **OK** on the Confirm Delete dialog. To close the dialog without deleting the conservation program, click **Cancel**.

Activating Conservation Programs

To activate an inactive or pending conservation program, click **Activate**.

Deactivating Conservation Programs

To deactivate an active or pending conservation program, click **Deactivate**.

Setting the Status of Conservation Programs to Pending

To set the status of an active or inactive conservation program to Pending, click **Pend**.

Rebate Definition Zone

You use the Rebate Definition zone to add, view, and edit rebate definitions associated with a conservation program. A rebate definition defines the types of rebates allowed for a specific conservation program. The Rebate Definition zone displays the following details for each claim line:

- Rebate Definition
- Status (Active or Inactive)
- Rebate Amount per Unit
- Icons to Edit, Delete, and Activate/Deactivate rebate definitions

Adding Rebate Definitions

Use the following procedure to add a rebate definition:

- Click **Add** in the title bar of the Rebate Definition zone.
- Enter a Description of the rebate definition.
- Select the **Product** for the rebate definition from the dropdown list.
- Select the **Service Type** for the rebate definition from the dropdown list.
- Enter the **Rebate Per Unit Amount** for the rebate definition.
- Enter the **Rebate Unit of Measure** for the rebate definition.
- Select the **Expense Adjustment Type** for the rebate definition from the dropdown list.
- Enter the **Presumed Energy Savings** for the rebate definition.
- Select the **Presumed Energy Savings** Unit of Measure for the rebate definition from the dropdown list.
- Enter the **Manufacturer** and **Model** for each item eligible to be submitted as a rebate claim line.
- To upload manufacturer/model information from a comma-separated-values file, click **CSV File to Upload**.
 - Click **Browse** on the File Upload dialog, and browse to the file to be uploaded.
 - Click **Upload**.

- Click **Save**. To return to the Rebate Definition zone without adding the rebate definition, click **Cancel**.

Editing Rebate Definitions

Use the following procedure to edit a rebate definition:

- Click the edit icon for the rebate definition you wish to delete.
- Edit the details of the claim line as appropriate.
- Click **Save**. To return to the Rebate Definition zone without changing the rebate definition, click **Cancel**.

NOTE:

You can edit a rebate definition only when it is Active.

Deleting Rebate Definitions

Use the following procedure to delete a rebate definition:

- Click the delete icon for the rebate definition you wish to delete.
- Click **OK** on the Confirm Delete dialog. To close the dialog without deleting the rebate definition, click **Cancel**.

Activating Rebate Definitions

To activate an inactive rebate definition, click the **Activate** button for the rebate definition you wish to activate.

NOTE:

You can only activate rebate definition that is currently Inactive.

Deactivating Rebate Definitions

To deactivate an active rebate definition, click the **Deactivate** button for the rebate definition you wish to deactivate.

NOTE:

You can only deactivate rebate definition that is currently Active.

Rebate Claim Statistics Zone

You use the Rebate Claim Statistic zone to view statistics for submitted rebate claims based on the current conservation program. This zone displays the following:

- Conservation Program
- Statistics Date/Time
- Statistics Charts

Statistics Graphs

This zone displays statistics for submitted rebate claims using the following charts:

- **Statistics by Status:** A pie chart that displays the number and percentage of claims for each status.
- **Statistics by Month:** A line chart that displays the number of claims completed each month.
- **Statistics by Product:** A pie chart that displays the total value, percentage, and number of claim lines completed for each product.
- **Statistics by Claim Age:** A line and bar chart that displays the number of new claims per month (line graph) and the number of claims per age grouping (bar graph).

To refresh statistics, click **Refresh Statistics** on the title bar of the Rebate Claim Statistics zone.

Configuration

This section contains additional configuration topics for Customer Care and Billing.

Configuring Zones

Many zones in Oracle Utilities Customer Care and Billing do not require configuration by your implementation team. For example, the base package is shipped with the Account Financial History zone that appears on the Control Central - Account Information portal. This zone does not require configuration because its zone type has no configurable options (i.e., its behavior is static).

Other zones require configuration before they can be used because their behavior is dynamic. The topics in this section provide tips and techniques on how to configure zones in Oracle Utilities Customer Care and Billing.

FASTPATH:

Refer to [The Big Picture of Portals and Zones](#) in the *Oracle Utilities Application Framework Administration Guide* for a description of portal and zone functionality.

Configuring Timeline Zones

A timeline zone is a zone that may be configured on the account or customer information tabs of Control Central. This type of zone contains one or more "lines" where each line shows when significant events have occurred. For example, you can set up a timeline zone that has two lines: one that shows when payments have been received from a customer, and another that shows when bills have been sent to the customer.

FASTPATH: For a complete description of the numerous features available on a timeline zone, refer to [Timeline Zone](#).

Configuration is required to enable a timeline zone. A timeline zone is made up of one or more algorithms where each algorithm returns "events" for a certain type of record (payment or bill for example). The product delivers several algorithm types for the timeline zone to include "events" for many different transactions in the product. Each algorithm type uses the Account or Person in context to retrieve the related information.

- Refer to [Configuring Timeline Zones](#) in the framework portion of the administration guide for information about the type of detail each event may return. The base provided algorithm types include many parameters to allow the implementation to configure much of this information as desired.

- Click [here](#) to see the algorithm types available for this plug-in spot.

Many of the algorithms support supplying a [BPA script](#) to launch when a user clicks on an event on a timeline. Be sure to review each algorithm type to see what types of events support a BPA script. For each event, determine if your business practice warrants the development of a BPA script.

After configuring the algorithms per your business needs, the following additional steps are needed:

- Set up a [zone](#) that references these algorithms. The zone will reference the **F1-TIMELINE** zone type.
- Link the zone to the appropriate portal(s) (e.g., [Control Central - Account Information](#) or [Control Central - Customer Information](#)).
- Update your users' [portal preferences](#) and [security rights](#) so they can see the zone in the desired location on the portal(s).

You can set up many timeline zones. For example,

- You might want different zones to appear on a portal depending on the type of user. For example, you might want one timeline for billing clerks, and a different one for customer service representatives.
- For aesthetic reasons, you might want multiple simple timeline zones to appear on a given portal rather than one complex timeline zone.
- You might want to set up context specific timeline zones. For example, you might want to have one timeline zone that is premise-oriented and another that is person-oriented.

Configuration Migration Assistant (CMA) Addendum

This section is an addendum to the general [Configuration Migration Assistant](#) section in the *Oracle Utilities Application Framework Administration Guide*. This section assumes that you are familiar with the concepts of Configuration Migration Assistant (CMA).

NOTE: CMA is designed to migrate configuration data only. Since base product data should be updated only through product patching mechanisms, CMA cannot currently be used to migrate master/transactional data that contains system-generated primary keys, such as customer account or billing information.

This addendum describes Configuration Migration Assistant (CMA) functionality that is specific to Oracle Utilities Customer Care and Billing.

The following sections provide information about what is provided in the C1-owned base package:

Base Package Migration Plans

The C1-owned base package provides a large number of migration plans to support migrating configuration and/or administration data from one environment to another.

Use the following procedure to access the base package migration plans:

1. Navigate to **Admin > Implementation Tools > Migration Plan > Search**.
2. Enter “C1” in the **Migration Plan** field.
3. **Click Refresh**.
4. Select a migration plan in the search results list. The details of the selected migration plan are displayed in the Migration Plan portal.

NOTE: The C1-owned base package plans can also be used as a basis for custom migration plans. To create a custom migration plan, select a plan to base the custom migration plan on, click the Duplicate button, and define the custom

plan to meet the implementation's requirements as described in [Defining a Migration Plan](#) in the *Oracle Utilities Application Framework Administration Guide*.

Base Package Migration Request

The C1-owned base package provides two migration request to support migrating configuration and/or administration data from one environment to another.

Use the following procedure to access the base package migration requests:

1. Navigate to **Admin > Implementation Tools > Migration Request > Search**
2. Enter "C1" in the **Migration Request** field.
3. Click **Refresh**.
4. Click the **Copy and Sync Control Data** link in the search results list.

The details of the migration requests are outlined below. Please use the Migration Request portal to view additional details about this migration request.

Base Admin Data Migration Request

- **Migration Request:** C1-AdminBasic
- **Description:** Base Admin Data
- **Detailed Description:** Base Admin Data.
- **Migration Plans:** This migration request includes base package migration plans used by Oracle Utilities Customer Care and Billing.

Copy and Sync Control Data Migration Request

- **Migration Request:** C1-CopyControlTables
- **Description:** Copy and Sync Control Data
- **Detailed Description:** Copy and Sync Control Data.
- **Migration Plans:** This migration request includes base package migration plans used by Oracle Utilities Customer Care and Billing.

For additional information about these migration plans, see [Base Package Migration Plans](#).

NOTE: The C1-owned base package request can also be used as a basis for custom migration requests. To create a custom migration request, select the base package request, click the Duplicate button, and define the request as to meet the implementation's requirements as described in [Defining a Migration Request](#) in the *Oracle Utilities Application Framework Administration Guide*.

Wholesale and Piecemeal Migrations

There are two general types of migrations used with the Configuration Migration Assistant: wholesale migrations and piecemeal migrations.

Wholesale Migrations

Wholesale migrations are used when migrating all the configuration and/or administration data from one environment to another. For example, a wholesale migration might be used when migrating admin data from a development or test environment to a production environment. For more on this type of migration, see [Wholesale Migrations](#) in the *Oracle Utilities Application Framework Administration Guide*.

The following is a high-level overview of the steps involved when executing a wholesale migration.

1. Process the “F1-SchemaAdmin” (FW Foundation) migration request (This request contains migration plans for Field, Lookup, Char Type, Currency Code and FK Ref).
2. Process the “C1-AdminBasic” migration request. This includes copies of framework migration plans (including plans for Business Objects, Algorithms, and Feature Configurations) from the “F1-FrameworkAdmin” migration request, as well as independent base package C1-owned wholesale migration plans which can be run first.
3. Process any of the other delivered framework-based (F1-owned) migration requests as needed (except for the “F1-FrameworkAdmin” migration request which is already incorporated in #2)
4. Process the “C1-CopyControlTables” migration request. This includes base package C1-owned wholesale migration plans with dependency on “C1-AdminBasic” migration request.

Piecemeal Migrations

Piecemeal (or "non-wholesale") migrations are used when migrating a small portion (or piece) of configuration and/or administration data from one environment to another. For example, a piecemeal migration might be used when migrating groups and rules from a development or test environment to a production environment. For more on this type of migration, see [Piecemeal Migrations](#) in the *Oracle Utilities Application Framework Administration Guide*.

The C1-owned base package does not contain piecemeal migration plans for Oracle Utilities Customer Care and Billing.

Processing Notes for Specific Objects

The following limitations apply to certain Oracle Utilities Customer Care and Billing objects when using Configuration Migration Assistant:

- An Issuing Center object references a User. If this user does not exist in the target system, Configuration Migration Assistant cannot apply the requested changes.
- A Case Type object references an Application Service. If this service does not exist in the target system, Configuration Migration Assistant cannot apply the requested changes.
- Collection Agency and Service Provider objects reference a Person. If this person does not exist in the target system, Configuration Migration Assistant cannot apply the requested changes.
- Service Provider and Tender Source objects reference a Service Agreement. If this service agreement does not exist in the target system, Configuration Migration Assistant cannot apply the requested changes.
- If your migrateable object includes log tables, you may add your log entity to the characteristic type F1-MGO. You should also mark the log table as a “Non-Migrateable Table” in the Maintenance Object options.
- Transactions are applied in an unspecified order (and probably in order numerically by a randomly generated ID value). CMA only looks at "hard" constraints when determining what to put into the same transaction. Any "soft" constraints such as characteristics and algorithm parameters that might have FK references to other objects are not processed by CMA. Unless the migration plan ensures that related items go into the same transaction, they will end up in different ones, and must need to apply again and again until eventually all gets applied.

To Do Lists Addendum

This section is an addendum to the general [To Do Lists](#) chapter. This addendum describes the To Do functionality that is specific to Oracle Utilities Customer Care and Billing.

Assigning A To Do Role

As described in [To Do Entries Reference A Role](#), each To Do entry requires a role. To Do entries created in Oracle Utilities Customer Care and Billing may attempt to assign a role based on an account management group or division if it is applicable to the type of data related to the To Do entry.

As described in [The Big Picture of To Do Lists](#), users are informed that something requires their attention by entries that appear in a To Do List. For example, consider what happens when billing can't find a reading (and it's not allowed to estimate):

- The billing process creates a bill segment that is in error (meter read cannot be found).
- This bill segment that's in error, in turn, triggers the creation of a To Do entry.
- The To Do entry is assigned a role. A role is one or more users who can look at / work on the entry.
- When users view a To Do List, they only see entries addressed to roles to which they belong.

You can optionally use account management groups (AMG) to define the respective role to be assigned to To Do entries that are associated with an account and To Do type. For example, you can create an AMG called Credit Risks and assign this to accounts with suspect credit. Then, whenever an account-oriented To Do entry is created for such an account, it will be assigned a role based on the Credit Risks AMG. Refer to [Setting Up Account Management Groups](#) for more information.

By assigning an AMG to an account, you are telling the system to address this account's To Do list entries to the roles defined on the AMG (note, each To Do type can have a different role defined for it on an AMG).

You can optionally use division to define the respective role to be assigned to To Do entries that are associated with an account and To Do type. For example, you may have a division called California Operations and assign this to accounts located in California. Then, whenever an account-oriented To Do entry is created for such an account, it will be assigned a role based on the California Operations division. Refer to [Setting Up CIS Divisions](#) for more information.

A To Do Pre-Creation installation options plug-in is provided to determine the appropriate To Do Role for an account based on AMG and division setup. If plugged in, the logic to determine To Do role for an account is performed whenever a To Do entry is created. Refer to [CI-TDCR-DFRL](#) for further details on how this plug-in works.

FASTPATH:

Refer to [To Do Entries Reference A Role](#) for the details of how an initial role is assigned to To Do entries.

System To Do Types

NOTE:

List of available To Do types. The To Do types available with the product may be viewed in the [application viewer](#)'s [To Do type](#) viewer. In addition if your implementation adds To Do types, you may [regenerate](#) the application viewer to see your additions reflected there.

Background Processes Addendum

This chapter is an addendum to the general [Defining Background Processes](#) chapter. This addendum describes the background processes that are provided with Oracle Utilities Customer Care and Billing.

NOTE: List of system background processes. The list of background processes provided in the base product may be viewed in the [application viewer's batch control](#) viewer. In addition if your implementation adds batch control records, you may [regenerate](#) the application viewer to see your additions reflected there.

Batch Process Dependencies

The contents of this section illustrate the periodicity and dependencies between the various background processes described above.

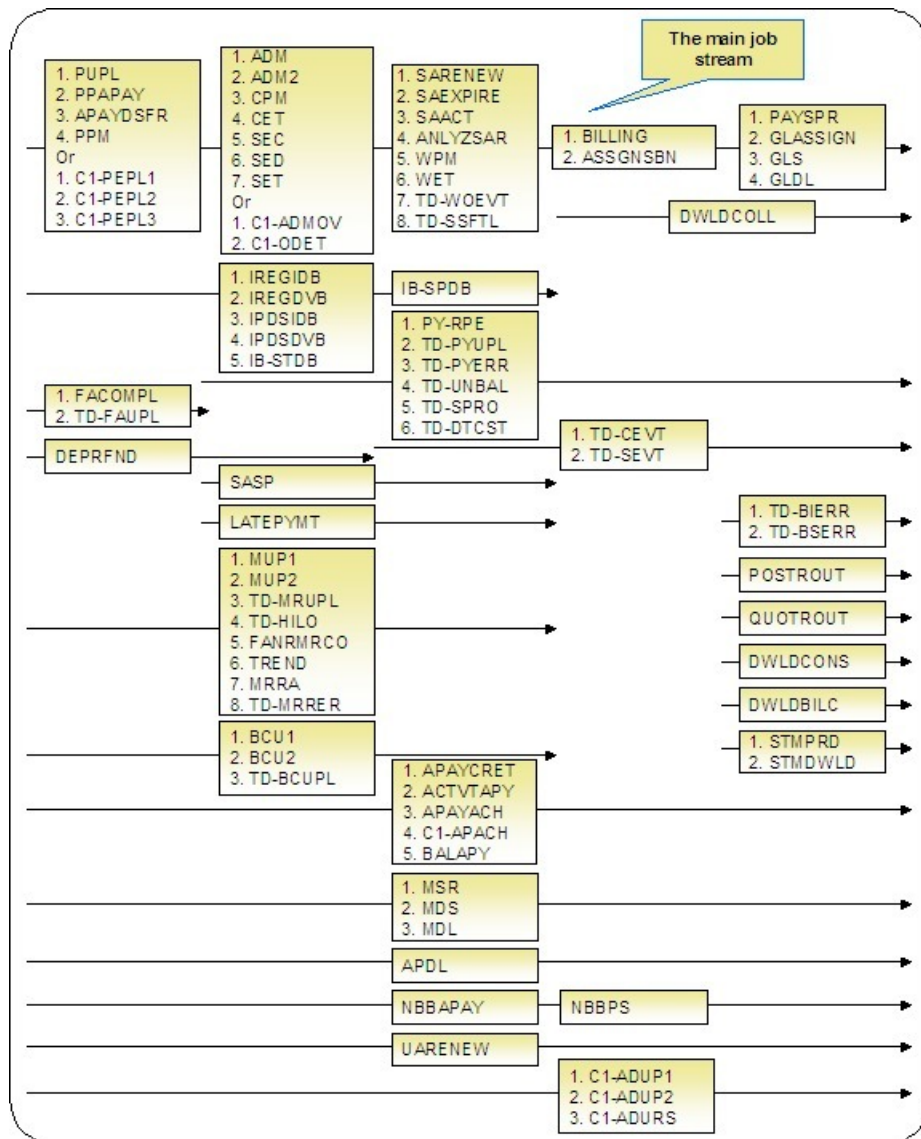
Batch Schedulers and Return Codes

If you use a batch scheduler (e.g., Control-M, Tivoli) to control the execution of your batch processes, it will be interested in the possible values of each process's return code. The return code is a number that indicates if the process ended successfully. All product processes will return one of the following return code values:

- 0 (zero). A value of zero means the batch process ended normally.
- 2. A value of 2 means the batch process detected a fatal error and aborted.

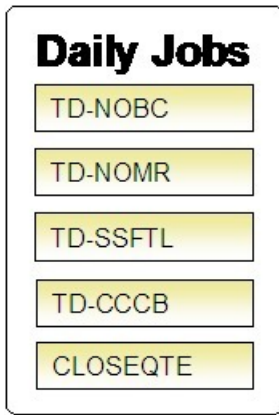
The Nightly Processes

The following diagram illustrates the dependencies between the batch processes.



The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially. When multiple boxes exist on a timeline, all processes in an earlier box must execute before the subsequent box is executed. Those timelines that appear beneath the Main Job Stream's timeline indicate when the timeline's respective processes can be executed in respect of the Main Job Stream.

The following diagram illustrates the daily batch processes for which there are no dependencies.



The mnemonics in the boxes refer to the individual batch processes described above.

NOTE:

No dependencies exist. As you can see, there are no dependencies between the boxes (meaning they may be run in parallel).

The Hourly Processes

The following diagram illustrates the dependencies between the hourly batch processes.



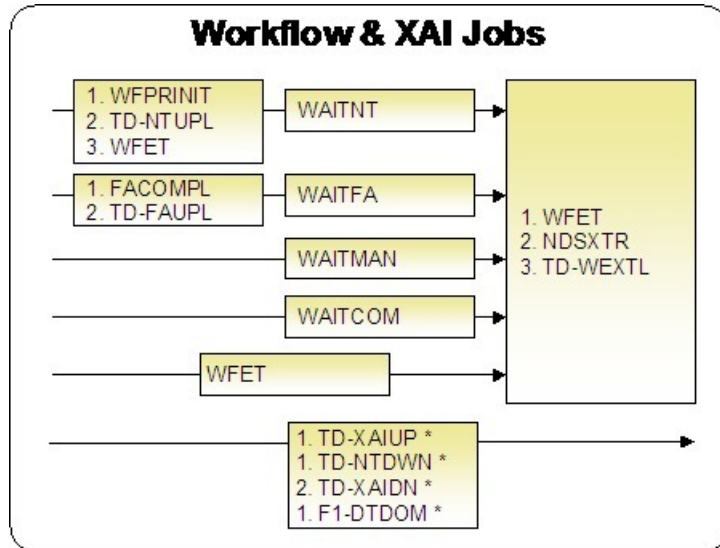
The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially.

NOTE:

No dependencies exist. As you can see, there are no dependencies between the boxes (meaning they may be run in parallel).

The Workflow and XAI Processes

The following diagram illustrates the dependencies between the workflow and XAI background processes. While these processes should be run at least once a day, you may want to consider running them more frequently (depending on how frequently you interface notifications and field activities into the system).



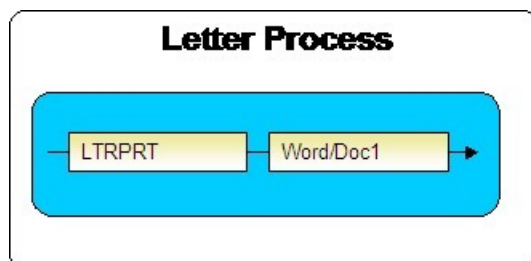
The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially. When multiple boxes exist on a timeline, all processes in an earlier box must execute before the subsequent box is executed.

* These processes create and/or clean up To Do entries for XAI upload staging, notification download staging, XAI download staging records or outbound messages in error. They are only applicable if your organization is using the XAI tool because only the XAI tool will mark one of these records in error.

The Letter Processes

To extract information for your various letters, only one background process, LTRPRT, is required regardless of the different types of letters you have. This process simply calls an algorithm plugged-in on the respective letter template to construct its flat-file content.

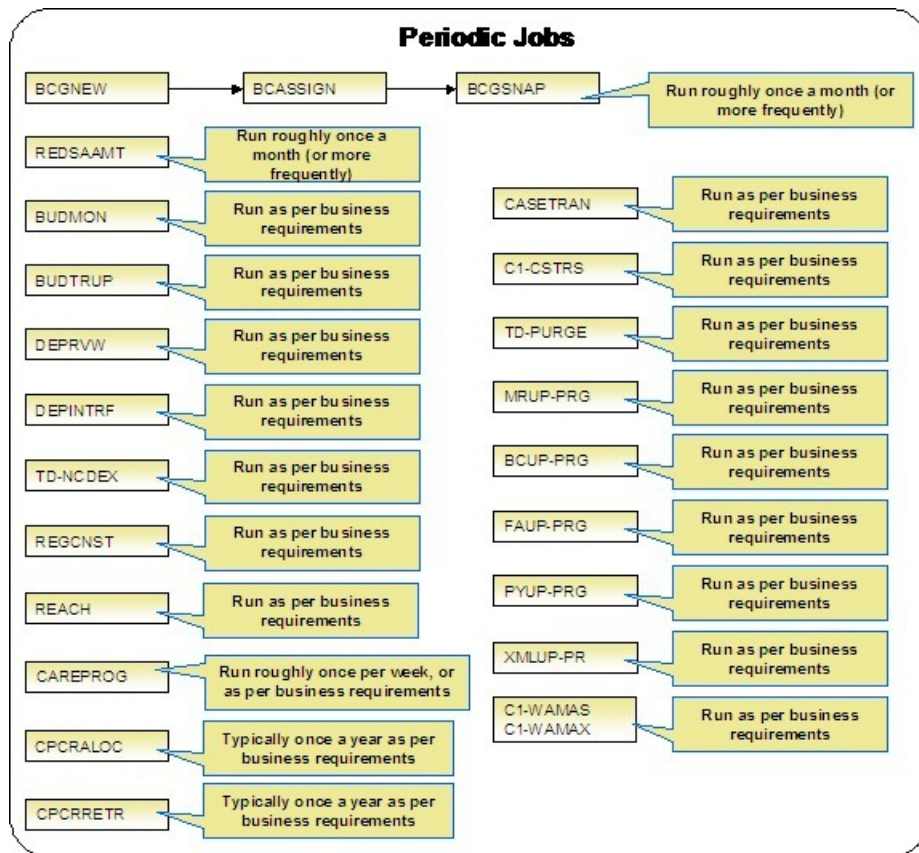
The following diagram illustrates the dependencies for the letter background process. While this process should be run at least on a daily basis, you may want to consider running it more frequently (depending on how frequently you produce letters).



The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially. When multiple boxes exist on a timeline, all processes in an earlier box must execute before the subsequent box is executed.

The Periodic Processes

The following diagram illustrates the dependencies between the periodic background processes. While many of these processes should be run at least on a monthly basis, you may want to consider running them more frequently (depending on business requirements).



The mnemonics in the boxes refer to the individual batch processes described above.

NOTE:

Few dependencies exist. As you can see, there are few dependencies between the boxes (meaning they may be run in parallel).

How To Set Up A New Extract Processes

Several background processes delivered with the system are used to interface information out of the system. The topics in this section describe when and how to introduce an additional extract process.

Setting Up Meter Read Extracts

You will need a meter read extract for every mechanism your company uses to route meter read requests to the software that handles your meter reading requests. For example:

- You will need a meter read extract to interface records to your handheld device software. The MDL process delivered with the system is intended to be used to handle this function. This process will have to be populated to format the output records in keeping with the needs of your meter reading software.
- If you interface some meter read requests to automatic meter reading software, you will need a new meter read extract process.

If you need additional meter read extract processes, set up the following information:

- Add a new [batch control](#) record. Populate the fields as follows:
- **ID.** Assign an easily recognizable unique ID for the meter read extract process.
- **Description.** Enter a description of the meter read extract process.
- **Accumulate All Instances.** Turn this switch on.
- Use [Route Type](#) to define the meter read extract process to be used for each route type.

NOTE:

Route types are defined for each route linked to every service cycle. Refer to [Setting Up Service Cycles And Routes](#) for more information.

- It may be necessary to register the process with your scheduler software.

Setting Up Automatic Payment Extracts

You will need an automatic payment extract for every mechanism your company uses to route automatic payment requests to a financial institution / clearing house. For example:

- You will need an automatic payment extract to interface records to the Automated Clearing House (ACH) if you allow customer to pay via credit card or direct debit from a checking account. The **APAYACH** and **C1-APACH** processes delivered with the system are intended to be used to handle this function.

If you need additional automatic payment extract processes, set up the following information:

- Add a new [batch control](#) record. Populate the fields as follows:
- **Batch Process.** Assign an easily recognizable unique ID for the automatic payment extract process.
- **Description.** Enter a description of the automatic payment extract process.
- **Accumulate All Instances.** Turn this switch on.
- Use [Auto Pay Route Type](#) to define the auto pay extract process to be used for each route type.

The Big Picture of Sample & Submit

Sample and Submit refers to the ability to create Activity Requests. This is functionality that enables an implementer to design an ad-hoc batch process using the configuration tools.

Some examples of such processes are:

- Send a letter to customers that use credit cards for auto pay and the credit card expiration date is within 30 days of the current date.
- Stop auto pay for customers that use credit cards as the form of payment if the credit card has already expired. Notify the customer that their auto pay agreement has been terminated and that they need to call to reinstate.
- Select auto pay accounts that have more than X non-sufficient fund penalties, stop the auto pay agreement and notify the customer.

NOTE:

The terms *activity request* and *sample & submit request* may be used interchangeably.

Activity Type Defines Parameters

For each type of process that your implementation wants to implement, you must configure an activity type to capture the appropriate parameters needed by the activity request.

Preview A Sample Prior To Submitting

To submit a new activity request, a user must select the appropriate activity type and enter the desired parameter values, if applicable.

After entering the parameters, the following actions are possible

- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can **Save** to submit the request, go **Back** to adjust the parameters or **Cancel** the request.
- Click **Cancel** to cancel the request.
- Click **Save** to skip the preview step and submit the request.

When an activity request is saved, the job is not immediately submitted for real time processing. The record is save in the status Pending and a monitor process for this record's business object is responsible for transitioning the record to Complete.

As long as the record is still Pending, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

The actual work of the activity request, such as generating customer contact records to send letters to a set of customers, is performed when transitioning to Complete (using an enter processing algorithm for the business object).

Credit Card Expiration Notice

The base product supplies a sample process to find customers that use credit cards for auto pay and the credit card expiration date is within x days of the current date.

To this functionality the following configuration tasks are needed:

- Define an appropriate [customer contact class](#) and [type](#) to use.
- Define appropriate activity request Cancellation Reasons. Cancellation reasons are defined using a customizable [lookup](#). The lookup field name is C1_AM_CANCEL_RSN_FLG.
- Define an activity type for the business object C1-NotifyExpiringCreditCardTyp . You may define default parameter values for the number of days for expiration and customer contact class and type.

Exploring Activity Request Data Relationships

Use the following links to open the application viewer where you can explore the physical tables and data relationships behind the activity request functionality:

- Click [C1-ACM-ACTTY](#) to view the activity type maintenance object's tables.
- Click [C1-ACM-ACTRQ](#) to view the activity request maintenance object's tables.

Defining a New Activity Request

To design a new ad-hoc batch job that users can submit via Sample and Submit, first create a new Activity Type business object. The base product BO for activity type C1-NotifyExpiringCreditCardTyp may be used as a sample.

The business object for the activity request includes the functionality for selecting the records to process, displaying a preview map for the user to review and to perform the actual processing. The base product BO for activity request C1-NotifyExpiringCreditCardReq may be used as a sample. The following points highlight the important configuration for this business object:

- Special BO options are available for activity request BOs to support the [Preview Sample](#) functionality.
 - Activity Request Preview Service Script. This script is responsible for retrieving the information displayed when a user asks for a preview of a sample of records.
 - Activity Request Preview Map. This is the map that is invoked to display the preview sample results.
- The enter algorithm plugged into the Complete state is responsible for selecting all the records that satisfy the criteria and processing the records accordingly.

Setting Up Activity Types

Activity types define the parameters to capture when submitting an activity request via Sample and Submit. To set up an activity type, open **Admin > Customer > Activity Type > Add**.

The topics in this section describe the base-package zones that appear on the Activity Type portal.

Activity Type List

The Activity Type [List zone](#) lists every activity type. The following functions are available:

- Click a [broadcast](#) button to open other zones that contain more information about the adjacent activity type.
- Click the Add link in the zone's title bar to add a new activity type.

Activity Type

The Activity Type zone contains display-only information about an activity type. This zone appears when an activity type has been broadcast from the Activity Type List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click the **Edit** button to start a business process that updates the activity type.

- Click the **Delete** button to start a business process that deletes the activity type.
- Click the **Duplicate** button to start a business process that duplicates the activity type.
- State transition buttons are available to transition the activity type to an appropriate next state.

Please see the zone's help text for information about this zone's fields.

Maintaining Sample & Submit Requests

Use the Sample and Submit transaction to view and maintain pending or historic activity requests. Navigate using **Menu > Batch > Sample & Submit Request > Search**.

Sample & Submit Request Query

Use the [query portal](#) to search for an existing sample & submit request. Once a request is selected, you are brought to the maintenance portal to view and maintain the selected record.

Sample & Submit Request Portal

This portal appears when a sample & submit request has been selected from the Sample & Submit Request Query portal. The topics in this section describe the base-package zones that appear on this portal.

Sample & Submit

The Sample & Submit zone contains display-only information about an activity (sample & submit) request. The following functions are available:

- Click the **Edit** button to modify the parameters. Refer to [Preview A Sample Prior to Submitting](#) for more information.
- If the activity request is in a state that has valid next states, buttons to transition to each appropriate next state are displayed.

Please see the zone's help text for information about this zone's fields.

Sample & Submit Log

This is a standard [log zone](#).

Defining Batch Schedule Options

The topics in this section describe how to set up the system to periodically execute batch job streams.

NOTE:

The batch scheduler is optional. Setting up the batch scheduler is only necessary if your organization uses the product's batch scheduler. If your organization uses a third party tool to manage the periodic execution of batch jobs, this section is not applicable.

Separate module. Batch scheduler functionality is associated with separate Workflow Scheduling module. If this module is not applicable to your business you may turn it off. Refer to [Turn Off A Function Module](#) for more information.

The Big Picture of Scheduling Batch Jobs

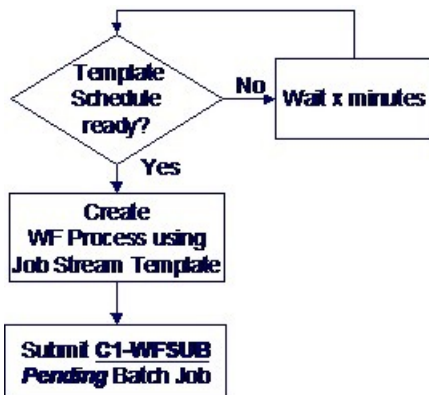
The following points provide an overview of how batch jobs are scheduled to run periodically.

Technical Implementation Of The Batch Scheduler

The topics in this section provide information about how your job streams are executed. This section is intended for your technical personnel who are responsible for configuring the program that periodically submits batch jobs.

A Program Determines If There's Work To Do

A program runs in the background looking for job stream (workflow process) templates whose [schedule](#) is ready to be processed. The following flowchart provides a schematic of its logic:



When C1-WFSUB executes, it causes the workflow events to activate. The activation plug-in on these workflow events creates Pending Batch Jobs

The following points summarize important concepts illustrated in the flowchart:

- When the job stream template [schedule](#) indicates it's time to start a job stream, a workflow process is created by copying the event types from the job stream (workflow process) template.
- In addition, a Pending batch job that activates the workflow process's events is created (i.e., the C1-WFSUB batch job is submitted).
- When C1-WFSUB executes, it activates the workflow process's events. The activation plug-in of these workflow events creates the job stream's Pending batch jobs. In addition, the activation algorithm transitions the workflow events into the Waiting state (they are waiting for the batch job to complete).

NOTE:

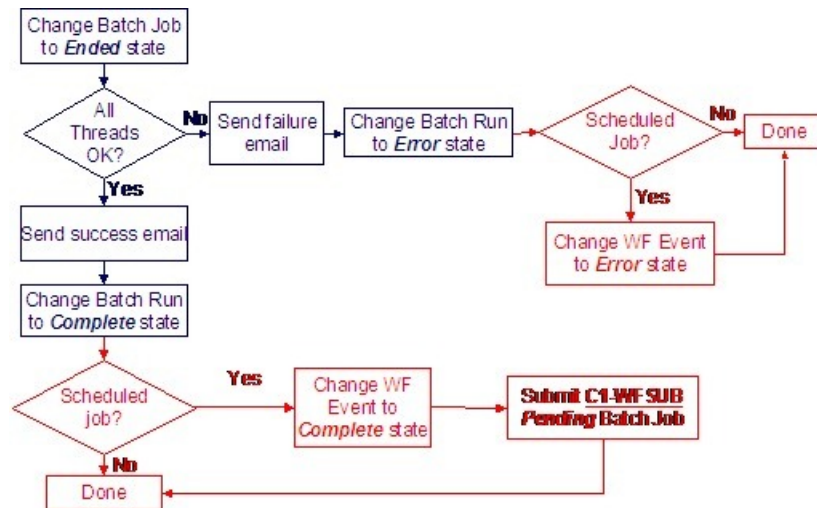
The identity of C1-WFSUB is not hard-coded. Rather, it's defined on the batch scheduler's [Feature Configuration](#).

NOTE:

Number of Threads. For batch jobs related to a job stream, the number of parallel threads to execute is defined on a parameter on the workflow event's activation algorithm.

Completing A Batch Program That's Part Of A Job Stream

When a batch program completes, a common routine is called. The following flowchart provides a schematic of this routine's logic where the logic applicable to the batch scheduler is highlighted in red.



The following points summarize important concepts regarding the batch scheduler logic illustrated in the flowchart:

- If at least one thread fails and this batch job is one related to a scheduled job stream, the status of the corresponding workflow event is changed to Error.
- If all threads are successful, and this batch job is one related to a scheduled job stream,
 - The status of the corresponding workflow event is changed to Complete.
 - In addition, a Pending batch job is submitted to activate workflow events that are dependent on the completion of a batch job. When this job executes, the "downstream" batch jobs will be submitted (and then the logic shown above starts again).

NOTE:

Batch Run Number. The routine also creates a characteristic for the workflow event to capture the batch run number associated with this batch run. The characteristic type to use is defined on the [feature configuration](#).

Polling Frequency

Usage of the batch scheduler requires that your system administrator create a job in the operating system to periodically execute the program described above. It is important to schedule the execution of this program very frequently (e.g., every five minutes) in order to avoid lengthy time gaps between the completion of a batch job and the triggering of the dependent jobs.

Job Stream, a Definition

The term "job stream" refers to a suite of batch jobs that run periodically. Most organizations have multiple job streams. For example,

- You'll have a job stream that contains batch jobs that run nightly
- You'll have a different job stream that contains the hourly batch jobs
- You'll have a different job stream that contains the batch jobs that extract data for your data warehouse
- Etc.

FASTPATH:

Refer to [Batch Process Dependencies](#) for a description of sample job streams.

A Workflow Process Is Created Each Time A Job Stream Executes

The system creates a [workflow process](#) each time a job stream executes. The workflow process has a separate workflow event for each batch job in the job stream. The batch jobs are submitted when the workflow process's events are activated (i.e., each workflow process event submits a specific batch job).

A Workflow Process Template Defines The Batch Jobs In A Job Stream

The system creates a job stream's workflow process using a [workflow process template](#). This means that a separate workflow process template exists for each job stream. For example, there is a workflow process template for the nightly job stream and another for the weekly job stream, etc.

There are typically dependencies between a job stream's batch jobs. For example, the billing batch job might be dependent on the successful execution of the payment upload batch job. Dependencies between batch jobs are defined by setting up workflow event dependencies on the workflow process. For example, if the billing batch job should only run after the payment upload batch job completes, you'd set up the workflow event that submits the billing job to be dependent on the completion of the event that submits the payment upload job. Note, you can define multiple dependencies between batch jobs (i.e., you can indicate that both the meter read upload and payment upload must complete before billing starts).

Once you define your job streams using workflow process templates, you indicate your job streams in the batch scheduler [feature configuration](#). The [job stream summary](#) page and the [job stream creation schedule](#) use this information to display the appropriate job stream templates.

How To Start A Job Stream

You can manually start a job stream using the [Job Stream Summary](#) page.

Optionally, you can [set up a schedule](#) defining when the system should start a job stream (i.e., create a workflow process). For example, you can set up a schedule for the "nightly" workflow process template to indicate that a workflow process should be created every night at 5:30 pm, Monday through Friday.

- You can monitor the status of your job streams on the [Job Stream Summary](#) page.
- You can monitor the status of a specific execution of a job stream on the [Job Stream Details](#) page. This page shows the status of the events on a workflow process and a summary of the execution status of each event's batch job.

FASTPATH:

Refer to [Technical Implementation Of The Batch Scheduler](#) for more information.

Activating A Workflow Event Causes A Batch Run To Be Submitted

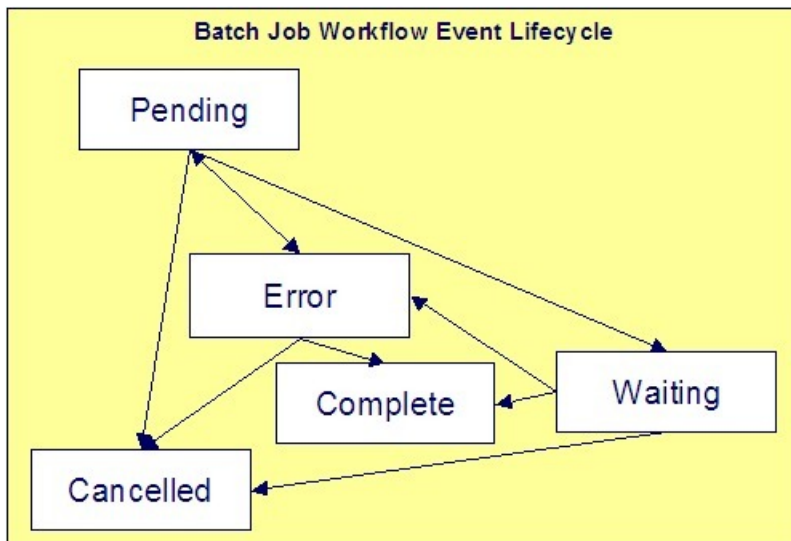
When a workflow event is activated, its activation algorithm creates a Pending batch run request. To be specific - the workflow event's Activation algorithm defines the batch process and the [number of parallel threads](#) that should be executed when the batch job is submitted. Please see the activation algorithm's parameters for a description of exactly how the Pending batch run request is created.

NOTE:

What activates workflow events? A batch process exists that is responsible for activating the workflow events associated with your job streams' workflow processes. This batch process is known as [C1-WFSUB](#). In addition, when this process is submitted, it only activates the workflow events associated with a specific workflow process template (the workflow process template is identified by a parameter supplied to C1-WFSUB). You may wonder why the standard WFET batch process (i.e., the standard workflow event activator) doesn't process the batch scheduler workflow events. The reason is that C1-WFSUB is executed many times during the processing of your job streams and we didn't want to clutter up the run statistics for WFET with the myriad executions of C1-WFSUB.

Workflow Event Status Reflects The Status Of The Batch Run

The system creates a [workflow process](#) each time a job stream executes. The workflow process has a separate workflow event for each batch job in the job stream. The batch jobs are submitted when the workflow process's events are activated (i.e., each workflow process event submits a specific batch job). The following diagram shows the potential state of these workflow events:



Workflow Event Lifecycle

The following points explain the relationship between a workflow event's status and the state of the corresponding batch job that it submits:

- Workflow events are initially created in the Pending state. The event's batch job has not been submitted when it's in this state.

- When the workflow event is activated, its activation algorithm attempts to submit a request to execute a batch job:
 - If there is something wrong with the activation algorithm's parameters, the event will enter the Error state. If this happens, you can:
 - Resubmit the batch job by changing the event's state back to Pending.
 - Cancel the batch job by changing the event's status to Canceled .
 - Skip this batch job by changing the event's status to Complete . Do this if the subsequent dependent batch jobs should proceed despite these errors.
 - If the batch run is submitted successfully, the workflow event enters the Waiting state (it is waiting for the batch job to complete).
 - When the batch job completes, the workflow event transitions into either the Complete or Error state:
 - If the batch run aborts due to too many errors, it transitions into the Error state. If this happens, you can:
 - Restart a batch job that aborted by changing the event's status back to Pending.
 - Cancel the batch job by changing the event's status to Canceled .
 - Skip this batch job by changing the event's status to Complete . Do this if the subsequent dependent batch jobs should proceed despite these errors.
 - If the batch run doesn't abort, due to too many errors, it transitions to the Complete state.
 - You can Cancel a Pending event if you don't want the batch job to be submitted.
 - A Pending event will be Canceled automatically by the system if the workflow process is canceled by a user.
 - A Waiting event will be Canceled automatically by the system if the workflow process is canceled by a user.

NOTE:

You can monitor the status of a workflow process's events and their related batch jobs on the [Job Stream Details](#) page.

Setting Up The Batch Scheduler

The following topics summarize how to set up the system to enable the scheduling of job streams.

Create A Workflow Event Type - Activation Algorithm For Every Batch Job

As described under [The Big Picture of Scheduling Batch Jobs](#), the system creates a workflow process when a job stream is initiated. The workflow process's events cause Pending batch jobs to be created when the workflow events are activated.

A base-package Workflow Event Type - Activation algorithm exists that creates a Pending batch job (the indtity of the batch job is defined as a parameter on the algorithm). This means that you need to set up a Workflow Event Type - Activation Algorithm for every batch job.

Besides defining the batch job's [batch control](#) code, the base-package algorithm, [C1-WFAC-CRBJ](#) also allows you to define the following values on each algorithm:

- **The number of parallel threads that should be submitted.** If this parameter is left blank, the batch job is submitted using a single thread. If a value greater than one is defined, the system automatically initiates this number of parallel threads. For example, if you want to kick the BILLING batch process using 20 parallel threads, specify a value of 20 in the respective algorithm's parameters. Refer to [Optimal Thread Count for Parallel Background Processes](#) for more information of executing a batch process in parallel threads.

- **The override email address used to inform that a batch job has finished.** If this parameter is specified, email is sent to this address when a batch job completes informing the recipient of its successful completion (or failure). Note, this address overrides the email address defined on the [Feature Configuration](#) for the batch scheduler (meaning that you need only populate this parameter if you want the email to be sent to someone other than the global email recipient defined on the Feature Configuration).
- **The value of all batch job-specific parameters.** Some batch jobs have job-specific parameters. For example, the Account Debt Monitor (ADM) batch process has a parameter that controls if calendar or workdays should be used when calculating dates. Values for the job-specific parameters may be defined on the batch code. In addition, override values may be defined on the algorithm's parameters. For example, you can define if calendar days or workdays are used when you set up the algorithm that submits the Account Debt Monitor batch process.

Create A Workflow Event Type For Every Batch Job

As described under [The Big Picture of Scheduling Batch Jobs](#), the system creates a workflow process when a job stream is initiated. The workflow process's events reference workflow event types. A separate workflow event type must be set up for every batch job (and you must reference the appropriate activation algorithm on each workflow event type).

Create A Workflow Process Template For Every Job Stream

As described under [The Big Picture of Scheduling Batch Jobs](#), the system creates a workflow process when a job stream is initiated. The system uses a workflow process template to create a workflow process. This means you must create a workflow process template for every batch job.

Set Up A Work Calendar

As described under [How To Start A Job Stream](#), you must set up a Job Stream Creation Schedule to define when each job stream should be started. You do this using the [Job Stream Creation Schedule](#) transaction. This transaction allows you to create the submission dates and times using a "recurrence schedule" (i.e., you don't have to type in 52 entries for a weekly job stream, you can rather have the system create the 52 entries for you). This feature uses a given [work calendar](#) to know what days are workdays and holidays. You must set up a work calendar (or reuse an existing work calendar) if you intend to use this feature.

Set Up A Job Stream Creation Schedule For Each Job Stream

As described under [How To Start A Job Stream](#), you must set up a [Job Stream Creation Schedule](#) to define when each job stream should be initiated. For example, you can set up a schedule to have your "nightly" job stream run at 5pm, Monday through Friday.

Set Up Characteristic Types

When the system activates a workflow event that submits a Pending batch job, it updates the workflow event with information about the batch job. It does this by populating characteristics on the workflow event. In order for the system to do this, you must set up the following characteristic types:

- Batch Job ID. This must be FK characteristic to the Batch Job object.
- Batch Control Code. This must be a FK characteristic to the Batch Control object.

- Batch Run Number. This must be an ad hoc characteristic.

In addition to the above workflow event characteristics, you must also set up a workflow process characteristic:

- Business Date. This must be an ad hoc characteristic. It holds the business date that's passed to the job stream's batch jobs (all batch jobs in a job stream use the same business date).

Set Up A User ID For Batch Jobs

When batch jobs execute, they must reference a given user's information for the following:

- Batch jobs can cause "auditable" events to occur. These events have a user ID associated with them.
- Some batch jobs access language-specific data, the language used is defined on a given user.
- When batch jobs complete, an email message is sent to a user's email address.

When a batch job is submitted, it references a specific user ID; this user ID is used for the above points. We recommend setting up a "dummy" user for this purpose (e.g., user ID = BATCH).

Set Up A Feature Configuration

After completing the above set up tasks, you must set up a [Feature Configuration](#) to provide the batch scheduler with the information it needs to operate.

The following points describe the various **Option Types** that must be defined for the feature configuration:

- Active Job Stream (WF Template Code). Set up a separate option for every job stream. The option's value is the Workflow Process Template associated with the job stream. Note, only these job streams appear on the [Job Stream Summary](#) page and are available in the [Job Stream Creation Schedule](#) page.
- Batch Code FK Char Type. Set up a single option to define the foreign key characteristic type used to reference the batch code on the workflow events. Refer to [Set Up Characteristic Types](#) for more information.
- Batch Code of WF Event Activation Process. Set up a single option to define the batch code of the batch job that is responsible for [activating the workflow events](#) that cause Pending batch jobs to be created.
- Batch Job ID FK Char Type. Set up a single option to define the foreign key characteristic type used to reference the batch job on the workflow events. Refer to [Set Up Characteristic Types](#) for more information.
- Batch Run Number Ad hoc Char Type. Set up a single option to define the ad hoc characteristic type used to reference the batch run's run number on the workflow events. Refer to [Set Up Characteristic Types](#) for more information.
- Business Date Ad hoc Char Type. Set up a single option to define the ad hoc characteristic type used to reference the batch run's business date on the workflow process (a single business date is used for all batch jobs initiated by a job stream). Refer to [Set Up Characteristic Types](#) for more information.
- Email Address on Batch Jobs. Set up a single option to define the email address used to notify a user when a batch job completes (note, the Email's subject line indicates if the batch job failed or succeeded). Note, you can override the email address on a specific batch job using a parameter on the Workflow Event Activation algorithm used to create the Pending batch job. If you leave this field blank (and no override is defined for the batch job), no email will be sent.
- SMTP Server Name and SMTP Port Number. If you want email sent informing of the completion of a batch job, define the SMTP server and port number.
- User ID on Batch Jobs. Set up a single option to define the user ID controlling user-oriented functionality. Refer to [Set Up A User ID For Batch Jobs](#) for more information.
- Work Calendar. Set up a single option to define the work calendar used when Workflow Process Recurrence Schedules are created. Refer to [Set Up A Work Calendar](#) for more information.

Maintaining Job Stream Creation Schedules

As described under [How To Start A Job Stream](#), you can set up a job stream creation schedule to define when a job stream (workflow process) should be automatically created by the system. Open **Admin > System > Job Stream Creation Schedule** to set up a job stream creation schedule.

Description of Page

On this page, you define when the system should automatically create a workflow process for a job stream template. To do this:

- Enter the **Job Stream Template**.

NOTE:

Only job streams defined on the [Feature Configuration](#) for the Batch Scheduler appear on this page.

- Enter the **Dates and Times** on which the system should automatically create the workflow process.

When a schedule record is added, the system sets its status to Pending.

Click the **Hold** button to change a Pending schedule to Hold if you don't want the system to create a workflow process on the date and time. You'd hold a record rather than delete it if you intend to release the hold in the near future.

Click the **Release** button to change a Held schedule to Pending if you want to release the hold.

When the system creates a workflow process for a job stream schedule record, the schedule record's status is changed to Process Created. There is one exception to this statement - if multiple Pending schedule records exist for a given workflow process template, the system will create a single workflow process for the one with the latest date / time and mark all others as Skipped Due To Overlap.

Schedule records in the Process Created and Skipped Due To Overlap states are protected.

Rather than entering the individual Dates and Times, you can press the **Recurrence** button to have the system generate these for you. Pushing this button causes the **Recurrence Window - Daily Pattern** to open.

The **Range From** and **To** define the dates during which recurring entries will be created.

The **Pattern** defines the frequency of recurrence:

- The Daily pattern will create a schedule for every date in the range defined above (subject to exceptions defined below). You can define the following for this pattern:
 - Use **Repeat Every** to define if the schedule should be created every day (a value of 1), every other day (a value of 2), every third day (a value of 3), etc.
 - Enter the **Start Time** defined on the schedule records.
 - Turn on **Include Weekends** if schedule records should also be created on weekends.
- The Hourly pattern will create a schedule for every hour in the date range defined above (subject to exceptions defined below). You can define the following for this pattern:
 - Use **Repeat Every** to define if schedule records should be created every hour (a value of 1), every other hour (a value of 2), every third hour (a value of 3), etc.
 - Enter the **Between** start time and end time to restrict the hours during which schedule records are created.
 - Turn on **Include Weekends** if schedule records should also be created on weekends.
- The Weekly pattern will create a schedule for every week in the date range defined above (subject to exceptions defined below). You can define the following for this pattern:
 - Enter the **Start Time** defined on the schedule records.

- Select the day(s) of the week that the schedule records should be created.

If You Work In A Non-English Language

The demonstration database is installed in English only. If you work in a non-English language, you must execute the [F1-LANG](#) background process on the demonstration database before using it as a Compare Source supporting environment. If you work in a supported language, you should apply the language package to the demonstration database as well.

If you don't execute F1-LANG on the demonstration database, any objects copied from the demonstration database will not have language rows for the language in which you work and therefore you won't be able to see the information in the target environment.

Security Addendum

This chapter is an addendum to the general [Defining Security and User Options](#) chapter. This addendum describes security functionality that is specific to Oracle Utilities Customer Care and Billing.

Implementing Account Security

CAUTION:

This section assumes you understand [The Big Picture of Row Security](#).

When you create an account, you must define which users can access the account's information. For example,

- If you have customers in two geographic territories, you may need to restrict access to accounts based on the office that manages each territory. For example, only users in the northern office may manage accounts in the northern territory.
- If you have industrial and residential customers, you may need to restrict access to these different customer segments based on the skill set of the users. For example, some users are skilled in dealing with industrial customers, while others are skilled in dealing with residential customers.

By granting a user access rights to an account, you are actually granting the user access rights to the account's bills, payment, adjustments, orders, etc.

FASTPATH:

Refer to [If You Do Not Practice Account Security](#) for setup instructions if your organization doesn't practice account security.

NOTE:

Account security may also affect persons and premises. Refer to [Persons Can Also Be Secured](#) for how access to person information is also restricted by account security. Refer to [Premises Can Also Be Secured](#) for how access to premise information is also restricted by account security.

The topics in this section describe how to implement account security.

Persons Can Also Be Secured

It's important to be aware that persons can also be secured as a result of "account security". It works like this:

- If a person is linked to at least one account, users will not be allowed access to the person (or the person's related information) unless they have access to at least one of the person's accounts.
- If a person is not linked to any accounts (a rare situation), any user may access the person.

NOTE:

How are persons linked to accounts? A person is linked to an account when an account is created using the methods described under [How To Add A New Customer From Control Central](#) and [Order User Interface Flow](#). In addition, you may manually link and unlink persons from account using the [Account - Person](#) page.

Premises Can Also Be Secured

It's important to be aware that premises can also be secured as a result of "account security". It works like this:

- If a premise is linked to at least one account, users will not be allowed access to the premise (or the premise's related information) unless they have access to at least one of the premise's accounts.
- If a premise is not linked to an account (a rare situation), then all users may access the premise.

NOTE:

How are premises linked to accounts? A premise is indirectly linked to an account. For the purpose of access restriction, we deem a premise as being linked to an account if at least one of its service points is linked to at least one of the account's service agreements.

Data Becomes Invisible When Access Is Restricted

The following summarizes the impact of a user not having access to an account:

- [Account Security and Control Central](#)
- [Account Security and Searches \(and Maintenance Pages\)](#)
- [Account Security and To Do Lists](#)

Account Security and Control Central

This section summarizes the impact of account security on [Control Central](#):

- Searches are affected as follows:
- An account will only be visible if a user has access to the account's access group.
- Persons that are not linked to accounts will be visible to all users.
- If a person is linked to an account, the person will only be visible if the user has access to at least one of the person's accounts.
- Premises that are not linked to accounts will be visible to all users.
- If a premise is linked to an account, the premise will only be visible if the user has access to at least one of the premise's accounts.
- The alerts that highlight the existence of "multiple relationships" are not impacted by account security. Specifically:

- The alert Person has multiple accounts will appear if the selected person is linked to multiple accounts, even if the user doesn't have access to every account. Note well, the person couldn't have been selected if the user didn't have access rights to at least one account.
- The alert Premise has multiple accounts will appear if the selected premise is linked to multiple account, even if the user doesn't have access to every account. Note well, the premise couldn't have been selected if the user didn't have access rights to at least one account.
- Only accounts to which the user has access will be displayed in the person tree.
- Only accounts to which the user has access will be displayed in the account tree.
- All other pages contain information related to Control Central's current account context. The current account context can never reference an inaccessible account and therefore these pages are not impacted by account security.

NOTE: If your implementation wishes to allow users to search for all accounts through Control Central without validating the user's access rights, this can be configured by setting up the **Search All Account** option type on the Customer Information Options [Feature Configuration](#) . If the user tries to select an account without having the required access, they will not be able to navigate to the Account Information tab on Control Central for the selected account.

Account Security and Searches (and Maintenance Pages)

Searches are the gateway to the information that appears on maintenance pages. In general, account-related information is suppressed when a user doesn't have access rights to the account. This suppression is true for rows that directly reference an account AND for rows that indirectly reference an account. For example:

- A user can only see bills associated with accounts to which they have access rights.
- A user can only see financial transactions associated with service agreements that are, in turn, associated with accounts to which they have access rights.

NOTE:

Person and premise searches are also impacted. Keep in mind that information will be suppress from both person and premise-oriented searches if the person / premise is related to accounts. Refer to [Persons Can Also Be Secured](#) for how access to person information is also restricted by account security. Refer to [Premises Can Also Be Secured](#) for how access to premise information is also restricted by account security.

Account Security and To Do Lists

Account security does NOT impact the information that appears in a user's To Do list. Rather, we have assumed that your To Do roles (and the users assigned to these roles) are consistent with your account security requirements. This can result in anomalies. For example, it's possible for a supervisor to assign a bill segment error to a user who doesn't have access to the bill segment's account. This user will then see the related To Do entry in their Bill Segments In Error To Do list. However, when they drill down on the entry, account security will manifest itself (i.e., the user won't be able to display the bill segment that's in error). This happens because the drill down causes the bill segment search logic to execute. This logic inhibits the selection of bill segments if the user can't access the related account.

To minimize these anomalies, we recommend the following:

- Setup [To Do Roles](#) consistent with your Data Access Roles.
- Setup [Account Management Groups](#) that are consistent with your Access Groups.
- Setup default To Do Roles on your Account Management Groups for each [To Do type](#).

Restricted Transactions

The following table lists all transactions that have some type of account security. The following notation is used to describe the type of account security:

- **Account-oriented.** This notation is used if the respective transaction uses basic account security (i.e., the user must belong to at least one data access role that has access to the account's access group in order to see the information).
- **Person-oriented.** This notation is used if the respective transaction uses person-oriented account security. Refer to [Persons Can Also Be Secured](#) for more information.
- **Premise-oriented.** This notation is used if the respective transaction uses premise-oriented account security. Refer to [Premises Can Also Be Secured](#) for more information.
- None of the above. Some unusual transactions have unusual implementations of account security. These are described below.

Transaction	Type of Account Security
Account	Account-oriented
Account Bill / Payment History	Account-oriented
Account Financial History	Account-oriented
Account Interval Info	Account-oriented
Account Payment History	Account-oriented
Account Person Replicator	Account-oriented
Account SAs for Debt Class	Account-oriented
Adjustment	Account-oriented
Adjustment Calculation Line Characteristics	Account-oriented
Appointment	Premise-oriented
Bill	Account-oriented
Bill Print Group	Person-oriented
Bill Segment	Account-oriented
Billable Charge	Account-oriented
Budget Review	Account-oriented
Case	Account-oriented, Person-oriented and Premise-oriented
Collection Agency Referral	Account-oriented
Collection Process	Account-oriented
Contract Option	Account-oriented
Control Central	Account-oriented, Person-oriented and Premise-oriented
Customer Contact	Person-oriented
Cut Process	Account-oriented
Declaration	Account-oriented
Deposit Review	Account-oriented
Field Activity	Premise-oriented
Field Order	Premise-oriented

Financial Transaction	Account-oriented
Financial Transaction on a Bill	Account-oriented
Financial Transaction on a Payment	Account-oriented
Interval Profile	Account-oriented (for SA-specific profiles)
Landlord Agreement	Account-oriented
Loan	Account-oriented
Match Event	Account-oriented
Multi-Cancel/Rebill	Account-oriented
Non-billed Budget	Account-oriented
Order	Account-oriented, Person-oriented and Premise-oriented
Overdue Process	Account-oriented
Pay Plan	Account-oriented
Payment	Account-oriented
Payment Arrangement	Account-oriented
Payment Arrangement for Bills	Account-oriented
Payment Event	The user must have access to ALL accounts linked to the payment event.
Payment Event Quick Add	The user must have access to ALL accounts linked to the payment event(s).
Payment Quick Add	Account-oriented
Payment / Tender Search	Account-oriented
Person	Person-oriented
Premise	Premise-oriented
Premise Management	Premise-oriented
Quote	Account-oriented
SA Billing History	Account-oriented
SA Cash Accounting Balance	Account-oriented
SA Financial History	Account-oriented
SA Relationship	Account-oriented
Service Agreement	Account-oriented
Service Credit Event	The user must have access to ALL accounts linked to the service credit membership that has the service credit event.
Service Credit Membership	The user must have access to ALL accounts linked to the service credit membership.
Service Provider SA Relationship	Account-oriented
Severance Process	Account-oriented
Start/Stop	Account-oriented
Statement	The user can see the statement if they have access to at least one account on the statement's statement construct.
Statement Construct	The user can see the statement if they have access to at least one account on the statement construct.

Terms of Service	Account-oriented (User must have access to at least one account linked to the SA collection in the TOS.)
TOU Map	Account-oriented (for SA-specific TOU maps)
Umbrella Agreement	Account-oriented (User must have access to at least one account linked to the SA collection in the UA's TOS.)
Write Off	Account-oriented
Write Off Process	Account-oriented

Account Security Case Studies

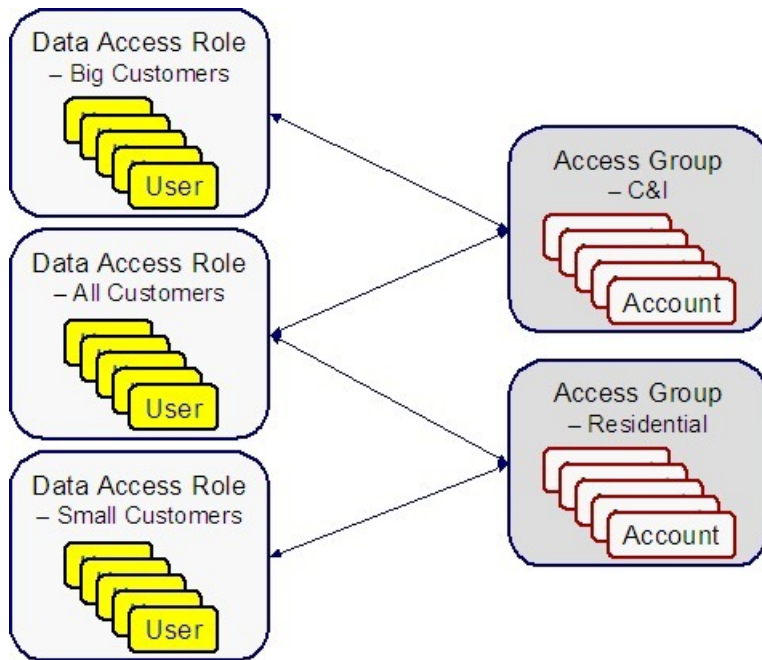
The topics in this section contain examples of how to implement account security. Use these examples to form an intuitive understanding of these objects. Once this intuition is obtained, you'll be ready to design the account security objects for your own company.

Securing Accounts Based On Customer Class

Assume the following security requirement exists:

- You have two broad groups of accounts:
 - Residential accounts.
 - Commercial / Industrial accounts.
- Users can be classified as have one of the following access rights:
 - May access all accounts.
 - May only access residential accounts.
 - May only access commercial / industrial accounts.

The following diagram illustrates the access groups and data access roles required to implement these requirements:



Notice the following about the above:

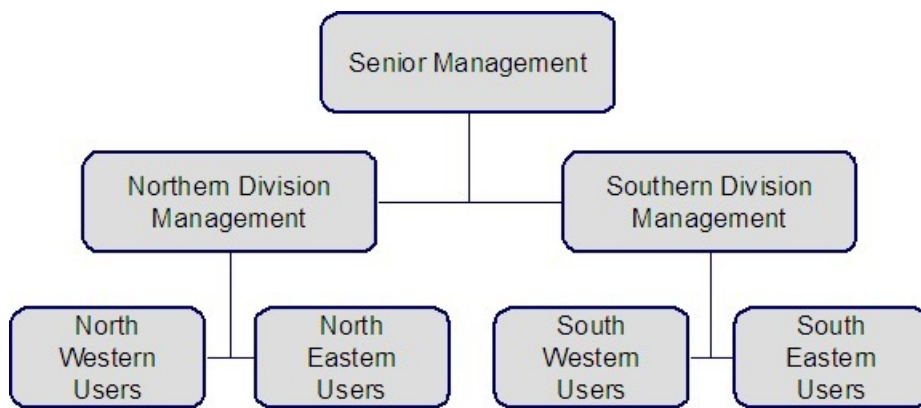
- There are 2 access groups because access to accounts is based on whether the account is considered to be residential or commercial/industrial.
- The Big Customers data access role is only linked to the C&I access group.
- The Small Customers data access role is only linked to the Residential access group.
- The All Customers access role is linked to both the C&I and Residential access groups. Users with this role can therefore access all accounts.

Securing Accounts Based On Region

Assume that accounts are classified as belonging to one of the following regions:

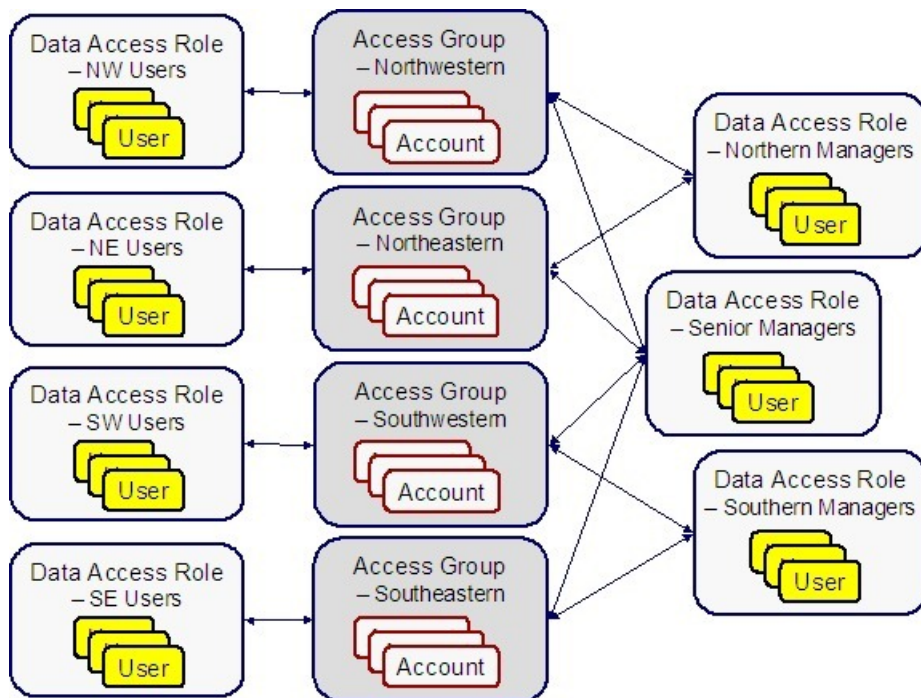
- Northwestern
- Northeastern
- Southwestern
- Southeastern

Assume the following company hierarchy exists:



- Senior Management has access to all customers
- Northern Division Management has access to all customers in the Northwestern and Northeastern divisions.
- Southern Division Management has access to all customers in the Southwestern and Southeastern divisions.
- Northwestern Users have access to all customers in the Northwestern division.
- Northeastern Users have access to all customers in the Northeastern division.
- Southwestern Users have access to all customers in the Southwestern division.
- Southeastern Users have access to all customers in the Southeastern division.

The following diagram illustrates the access groups and data access roles required to implement these requirements:



Notice the following about the above:

- There are 4 access groups because access to accounts is based on the region in which they are located (and there are 4 regions).
- There are 7 data access roles because each component of every layer of the access hierarchy requires a separate data access role.

The Default Access Group

There are two ways an access group can be assigned to an account:

- The base package will default an account's access group based on the user who adds the account. It uses the user's [default access group](#) to do this.
- If you can conceive of a rule to assign an appropriate access group to a newly added account, you can have your programming staff introduce a user exit to the account row program to implement this rule. For example, a user exit could be developed that assigns an access group to an account based on its customer class. The name of the user-exit is AFTER-MOVE-CORR-ADD and the name of the account row program is CIPCACCR.

WARNING:

Regardless of the method used to assign an account's access group, please be aware that the user who adds an account must have access to this access group.

NOTE:

Subsequent changes to an account's access group. A user may change an account's access group to any access group to which they have access.

If You Do Not Practice Account Security

If you do not restrict access to accounts (i.e., all users can access all accounts), you must set up one access group and one data access role and then indicate all users are part of this role. You should also define the access group as the default access group on all of your users (so that new accounts are all labeled with this access group).

Masking Sensitive Data

Refer to [Masking Data](#) for instructions describing how to configure the system to mask sensitive data like a customer's social security number or bank account number. If your implementation intends to mask any of the information that appears in the [Customer Information Zone](#) please navigate to this zone's documentation for special instructions.

Encrypting Sensitive Data

The product provides support for encrypting certain sensitive data, which may be necessary if your implementation wants to meet security standards such as those for credit card transactions, as specified by the Payment Card Industry Security Standards Council.

Please be sure that you are familiar with the information documented in [Database Encryption and Masking](#) before reading further.

The remaining details in this section provide information about the fields in the product that support encryption along with the suggested configuration for the **Encryption** feature configuration entry if your implementation opts to encrypt the data.

External Account ID and Credit Card Number Encryption

If your implementation would like to encrypt credit card IDs, the following entry should be added to the Encryption feature configuration so that the data stored in the External ID column on the Account Automatic Payment table is encrypted:

table='CI_ACCT_APAY', field='EXT_ACCT_ID', encryptedField='ENCR_EXT_ACCT_ID'

NOTE:

- The External Account ID field also exists on other tables such as Auto Pay Clearing Staging table (CI_APAY_CLR_STG) and the Payment Event Upload Staging table (CI_PEVT_DTL_ST). Feature configuration options do not need to be configured for the External Account ID field on these tables as the field on these related tables is encrypted based on the feature configuration specified above for the Account Automatic Payment table (CI_ACCT_APAY).
 - A customer's bank account ID or credit card number may also be stored in a column when an order is completed. A field encryption feature configuration for the Order Field table (CI_ENRL_FLD) does not need to be configured if the customer's bank account ID or credit card number is referenced on this table. The encryption for the field should be handled by a Column Reference - Preprocessing algorithm. The base product provides an Encrypt Account Auto Pay External Account Id Column Reference Value algorithm type for this purpose.
 - A field encryption feature configuration option needs to be configured for each schema field to be encrypted that represents a customer's bank account ID or credit card number. If the value of an encrypted schema field will later be populated on or compared against an encrypted table field, the schema field and the table field must share the same key alias.
-

Bank Account Number Encryption

If your implementation would like to encrypt the bank account number stored on the bank account table, the following entry should be added to the Encryption feature configuration:

table='CI_BANK_ACCOUNT', field='ACCOUNT_NBR', encryptedField='ENCR_ACCOUNT_NBR'

MICR ID Encryption

The product provides support for encrypting MICR ID associated with payments. In addition, the product supports capturing a hash representation of this field for searching purposes. This field is captured on three tables and entries should be added to the Encryption feature configuration for each as follows:

- **table='CI_PEVT_DTL_ST', field='MICR_ID', encryptedField='ENCR_MICR_ID', hashField='HASH_MICR_ID'**
- **table='CI_PAY_TNDR_ST', field='MICR_ID', encryptedField='ENCR_MICR_ID', hashField='HASH_MICR_ID'**
- **table='CI_PAY_TNDR', field='MICR_ID', encryptedField='ENCR_MICR_ID', hashField='HASH_MICR_ID'**

NOTE: All configurations must share the same key alias.

Person ID Number Encryption

The product provides support for encrypting an identifier associated with a person. In addition, the product supports capturing a hash representation of this field for searching purposes. Because the ID collection on Person includes an ID type, your implementation may decide to encrypt all IDs captured for the person or may choose to only encrypt IDs of one or more specific ID types. If you want to encrypt multiple ID types, you must configure a field encryption option type for each ID type. The following shows an example of an encryption feature configuration entry for encrypting an ID with the type "SSN":

table='CI_PER_ID', field='PER_ID_NBR', encryptedField='ENCR_PER_ID_NBR', hashField='HASH_PER_ID_NBR', where='ID_TYPE_CD=SSN'

NOTE:

- The Person ID Number also exists on the Order Person ID table (CI_ENRL_PER_ID) table. Feature configuration options do not need to be configured for the Person ID Number field on this table as the field is encrypted based on the feature configuration specified above for the Person Identifier table (CI_PER_ID).
-

- The Person ID Number may also be stored in a column when an order is completed. A field encryption feature configuration for the Order Field table (CI_ENRL_FLD) does not need to be configured if the Person ID Number is referenced on this table. The encryption for the field should be handled by a Column Reference - Preprocessing algorithm. The base package provides an Encrypt Column Reference Value algorithm type for this purpose.
 - A field encryption feature configuration option needs to be configured for each schema field to be encrypted that represents a Person ID Number. If the value of an encrypted schema field will later be populated on or compared against an encrypted table field, the schema field and the table field must share the same key alias.
-

Encrypting Legacy Data

If you enable encryption in an existing implementation, you should run the encrypt legacy batch controls to encrypt legacy data that persists in the database:

- F1-ENCRT: Encrypt Legacy Table Field Data
- F1-ENCRS: Encrypt Legacy Schema Field Data
- C1-ECRVL: Encrypt Legacy Order Field Col Ref Value

Before executing the batch processes, you must have a keystore file in the system to hold the keys for encrypting data and define an encryption feature configuration that contains the details for the fields you want to encrypt.

Defining Converted COBOL Program Options

The topics in this section describe the transaction that allows you to define the metadata for converted COBOL programs within the current environment's database.

CAUTION: Updating converted COBOL Programs requires technical knowledge of the system. This is an implementation and delivery issue and should not be attempted if you do not have previous experience.

Converted COBOL Program - Main

Use this transaction to define converted COBOL program user exits for your system. Navigate to this page using **Admin > Converted COBOL Program > Search**.

Description of Page

The following describes fields that are relevant to defining the user exit code that a converted COBOL Program should use:

Program Component ID represents the internal ID that is given to the converted COBOL program component.

Program Com Name is the physical name of the converted COBOL program component.

Template is the template used to generate the converted COBOL program component.

Location ID (*for development purposes*)

Table (*for development purposes*)

User Exit Program. Specify if you have written user exit code for this converted COBOL program component.

Development Status Flag. Options include Note Generated and Not Regeneratable.

Short Comments provides a short description of the converted COBOL program component.

Long Comments provides more details of the converted COBOL program component.

The **Variable Information** grid (including Variable Name, Occurrence, Variable Sequence, Variable Value, and Variable Alternate Value) is *for development purposes only*.

Integration

Oracle Utilities Customer Care and Billing provides tools to facilitate the integration other systems. This section provides technical information needed by your implementers to accomplish this.

CTI-IVR Integration

Oracle Utilities Customer Care and Billing provides tools to facilitate the integration with your Computer Telephony Integration/Interactive Voice Response (CTI/IVR) system. The interface provides the following functionality:

- The ability to launch Control Central for a particular account ID or phone number from an external application
- The ability to accept the next call, as dictated by the CTI software,
- The ability to perform an outbound phone call from within Oracle Utilities Customer Care and Billing

This document provides technical information needed by your implementers to fully integrate with your CTI/IVR system.

Launching The System From an External Application

The following section describe possible options to launch the system from an external system.

Launching The Application Using a URL

You launch the application using a URL. With this option you can set the system to launch a script upon startup. You can also indicate to the system to automatically load an appropriate page (if this information is not part of the script).

The application includes a simple sample html page that can launch CCB Control Central Search by Phone Number or by Central Search Account Id. The page is called CTISample.HTM and is located in /cm_templates.

FASTPATH:

Refer to [Launching A Script When Starting The System](#) for further information.

Receiving the Next Caller in the Queue

If your CTI-IVR system allows users to request the next caller waiting in a queue, the system provides a mechanism to integrate with this functionality.

A BPA script called **Get Next Caller (C1-CCByAcct)** is available that can be used to request the next call waiting in an inbound queue managed by a CTI application.

When the **Get Next Caller** BPA is executed, it launches a browser script function called **launchCTI** located in a file called **ext_cti.jsp**. The **launchCTI** function calls a function called **ctiGetNextCaller** to retrieve the next caller's account ID and uses [Launching A Script When Starting The System](#) to load Control Central for an account.

Customize Integration to Your Next Caller Function

The `ext_cti.jsp` file shipped with the base product provides sample functionality that should be replaced with the appropriate integration to your CTI application. In the sample provided, the `ctiGetNextCaller` randomly takes an account ID from a predefined list of accounts.

In order to integrate the next caller functionality with your CTI-IVR system, perform the following steps:

- Copy the JSP page `ext_cti.jsp` from the `/cm_templates` directory found under the web application root directory on your Oracle Utilities Customer Care and Billing server to the `/cm` directory.
- In the `/cm` directory, replace the contents of the `ctiGetNextCaller` function to retrieve the next caller ID from your CTI application.

Initiating an External Call

This section describes the automated dialer functionality provided with the system as well as information about integrating with your own automated dialer.

Overview of Automated Dialer

In order to initiate a call to a customer from within the system, a context menu item **Go To Automated Dialer** is available on the Person context menu. To call a customer displayed in the current context, choose this option from the person context menu and a window appears, showing a list of phone numbers defined for that person.

Select the desired phone number and click **Dial**.

NOTE:

Context Entry Secured. The [navigation key](#) for this window `automatedDialer` refers to an application service to facilitate application security. If your installation does not support an integration with external dialer software, configure the security settings to ensure that users do not have access to the application service for this context entry.

Technical Implementation of Automated Dialer

The popup window is implemented as a JSP page, which calls the JSP page `ext_cti_dialer.jsp` to integrate with an automated dialer. The `ext_cti_dialer.jsp` page provided with the system integrates to any soft phone protocol handler that launches a dialer based on overriding the “tel:” protocol from a browser, for example, Cisco IP Communicator.

Phone Dialer Configuration

If your implementation chooses to use the functionality provided with the system and integrate with a soft-phone dialer, you must copy the JSP page `ext_cti_dialer.jsp` from the `/cm_templates` directory found under the web application root directory on your Oracle Utilities Customer Care and Billing server to the `/cm` directory:

Customize Integration to Your Automated Dialer Software

In order to integrate with a different automated dialer software application, your implementers must modify the `ext_cti_dialer.jsp` to call the appropriate dialer.

- Copy the JSP page `ext_cti_dialer.jsp` from the `/cm_templates` directory found under the web application root directory on your Oracle Utilities Customer Care and Billing server to the `/cm` directory.
- Make the appropriate changes to the copy of `ext_cti_dialer.jsp` in the `/cm` directory to integrate with your automated dialer (e.g. change the protocol from “tel:” to “callto:”).

Customize Automated Dialer User Interface

Your implementation may choose to display a different user interface for the **Go To Automated Dialer** function than the one provided with the system. For example, perhaps there is more information that you would like to display in addition to the person's name and phone numbers. In order to do this, perform the following steps:

- Create your customized component to provide the desired functionality.
- Create a navigation key for your new component and indicate the URL being overridden. The remainder of the section walks you through these steps.

Go to **Utilities, System, Navigation Key** +.

For **Navigation Key**, specify a name for the new navigation key prefixed with CM.

For **URL Location**, select **External (Override)** to override a base navigation key.

When you select **External (Override)**, the **Overridden Navigation Key** becomes available. Select the **automatedDialer** navigation key because that is the key you are overriding.

The **URL Override** is the path on the web server to your custom component.

When overriding a navigation key, you must flush the system login cache on the web server. The navigation keys are stored in the system login cache, so the overrides do not become effective until the cache is flushed. To flush the cache, issue the following command in your browser's address bar: `http://server:port/flushSystemLoginInfo.jsp`, where server is the name or address of your web server and port is the port number of the application, for example, `http://CD-Implementation:7500/flushSystemLoginInfo.jsp`.

FASTPATH:

Refer to the [Defining Navigation Keys](#) for more information.

Analytics Integration Overview

Oracle Utilities Customer Care and Billing provides tools to facilitate the integration with Oracle Utilities Analytics.

Oracle Utilities Customer Care and Billing provides support for defining data load parameters, bucket configurations and characteristic mapping.

Refer to **Oracle Utilities Extractors and Schema for Oracle Utilities Customer Care and Billing** in the *Oracle Utilities Analytics Administration Guide* for detailed information on the data that are extracted out of Oracle Utilities Customer Care and Billing and loaded to the Oracle Utilities Analytics, along with the rules of data transformation.

Refer to *Oracle Utilities Analytics Dashboards for Customer Analytics, Revenue Analytics, and Credit & Collections Analytics* in the *Oracle Utilities Analytics Administration Guide* documentation for detailed information on how extracted and transformed data from Oracle Utilities Customer Care and Billing can be analyzed into actionable, useful applications.

Analytics Configuration

This section describes functionality provided for integrating with Oracle Utilities Analytics (OUA).

- **Analytics Configuration Portal**

Oracle Utilities Customer Care and Billing provides an Analytics Configuration portal that holds information on the all the analytics-oriented configuration activities. This portal provides a summary of how much configuration has been set up, and also provides links and guidelines for the areas that need configuration, at the minimum, to successfully extract data from Oracle Utilities Customer Care and Billing to OUA. The following provides more information about this functionality.

1. **Analytics-Oriented Master Configuration Details** – These data load parameters are used to extract, load, and transform data from Oracle Utilities Customer Care and Billing to Oracle Utilities Analytics. These parameters are used to identify and/or filter data when loading them into the data warehouse. Refer to the *Oracle Utilities Customer Care and Billing* section in the **Configuring Oracle Utilities Analytics** chapter of the *Oracle Utilities Analytics Administration Guide* for detailed information on each extract parameter.
2. **Bucket Configuration List** – This is a summary of the bucket configurations that are needed to classify and group extracted data within Oracle Utilities Analytics. The bucket configurations provide the ability to readily report and analyze data by buckets, or groups. Refer to the *Oracle Utilities Customer Care and Billing* section in the **Configuring Oracle Utilities Analytics** chapter of the *Oracle Utilities Analytics Administration Guide* for detailed information on each bucket configuration supported by Oracle Utilities Customer Care and Billing.

- **Characteristic Mapping**

Oracle Utilities Customer Care and Billing has identified the objects for which characteristic extraction is supported. This includes mapping the source characteristic entity to the target dimension, and defining which target columns are allowed to be mapped for which target dimension. The following table shows the supported mapping.

Object	Supported Target Dimension	Supported Target Columns
Account Characteristic	Account Dimension	User Defined Fields 6 - 15
Person Characteristic	Person Dimension	User Defined Fields 1 - 6
Premise Characteristic	Premise Dimension	User Defined Fields 6 - 18
Premise Characteristic	Address Dimension	User Defined Fields 7 - 16
Service Agreement Characteristic	Service Agreement Dimension	User Defined Fields 9 - 19

Self-Service Integration

The following sections describe functionality provided for the integration with self-service applications such as Oracle Utilities Customer Self Service (OUCSS).

About Self-Service Tasks

Self-service tasks hold information about self-service operations that customers initiate from an integrated self-service application, such as Oracle Utilities Customer Self-Service (OUCSS).

A few examples of self-service tasks are:

- One time payments
- Account verification
- Start / stop requests
- Auto pay setup

Refer to the Oracle Utilities Customer Self-Service (OUCSS) documentation for more details on supported self-service operations.

The following sections highlight self-service integration functionality.

Self-Service Task Type Defines Self-Service Task Behavior

Each type of self-service operation has specific processing requirements.

Self-service task type controls how a self-service task is processed. It defines the business object to use when creating the self-service task record. The business object defines the lifecycle of self-service task record.

The self-service task type can also specify common configuration data, such as settings for error handling.

Creating Self-Service Tasks

The customer self-service application invokes inbound web services to create self-service tasks. The system supplies an inbound web service for each supported self-service operation.

NOTE: For products that are continuing to use XAI for external messages, the product also includes XAI inbound services for most of the supported self-service task types. Note that the product recommendation is to discontinue use of XAI and use [inbound web services](#) instead.

Refer to the Oracle Utilities Customer Self-Service (OUCSS) documentation for more details on supplied inbound web services.

Each inbound web service references a service script that processes the specific self-service operation. Refer to the service script system data for processing details.

Setting Up Self-Service Task Configuration

The following topics highlight the general configuration steps required to use self-service task functionality and to integrate with a self-service application such as Oracle Utilities Customer Self-Service (OUCSS).

Self-Service Integration Configuration

The integration with a self-service application such as Oracle Utilities Customer Self Service (OUCSS) requires the setup of master configuration data that controls the processing of your self-service operations.

To set up self-service integration master configuration, navigate using **Admin > General > Master Configuration**. You are brought to an all-in-one portal with options to add/edit the Self-Service Integration master configuration record. Refer to the inline help for more details on how each section in this record is configured.

Defining Self-Service Task Types

A self-service task type defines properties that control how a self-service task is processed.

Refer to [About Self-Service Tasks](#) for an overview of business flag functionality.

To maintain the business flag types applicable to your product, open **Main > Self-Service > Self-Service Task Type**.

This is a standard [All-in-One portal](#).

The information captured on the self-service task type depends on the business objects supported by your product. Refer to the inline help text for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI_SVC_TASK_TYPE](#).

Maintaining Self-Service Tasks

This section describes the functionality supported for viewing and maintaining self-service tasks.

Refer to [About Self-Service Tasks](#) for an overview of self-service task functionality.

Navigate using **Main > Self-Service > Self-Service Task Search**. You are brought to a query portal with options for searching for self-service tasks.

Once a self-service task record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Self-Service Task** zone provides basic information about a self-service task. Refer to the inline help for more information.

Defining DataConnect Options

Oracle Utilities DataConnect facilitates data extraction from Oracle Utilities Customer Care and Billing for use in external applications; such as, analytics applications and energy management systems.

This section describes how Oracle Utilities DataConnect works and how to implement and configure the system to support data extract processing.

DataConnect Data Extracts

Oracle Utilities DataConnect can be used to extract two types of data:

- **Master Data**

(see [Master Data Extracts](#) for more information)

- Service agreement, account, and person related information
- Service point and premise related information
- Service point / meter history related information

- **Billing Data**

(see [Billing Data Extracts](#) for more information)

The extraction processes used for each type of data can be run to produce separate extracts or combined into one.

If individual flat files are preferred, the data can be correlated in external systems through service agreement and/or service point data that is included in each type of extract. All data extracts contain the following data elements that can be used for this correlation:

- **Service Point ID:** The service point ID in Oracle Utilities Customer Care and Billing and/or
- **Service Agreement ID:** The service agreement ID in Oracle Utilities Customer Care and Billing

External systems receiving data extracted from the Oracle Utilities DataConnect facilities can use these IDs to associate billing data to extracted master data.

If a single flat file is preferred, meaning an extract that contains billing data and master data in one export file, the Billing Data extraction process can be configured to retrieve the associated master data of the bills and place the information in the same row as the billing data.

Master Data Extracts

Master data extracts are performed through the use of data synchronization, audit algorithms, business objects, and related batch processes.

Batch processes are used to create initial load extracts for service point/premise and service agreement / account / person related master data. The initial load includes the historical data in the system (either all or a subset, depending on how the batch process is configured). Following the initial load, data synchronization requests are created when master data is added or changed.

Master Data Extracts: Maintenance Object Audit Algorithms

Detecting changes in master data can be identified by audit algorithms on the following maintenance objects:

- Service Agreement
- Account
- Person
- Service Point
- Premise
- SP/Meter History
- Meter

The Generic Change Data Capture algorithm type (F1-GCHG-CDCP), which is provided in the base package, can be used to create audit algorithms for following maintenance objects:

- Service Agreement
- Service Point
- SP/Meter History

In addition,

- An **Account Change Data Capture (SA-Based) – Include Reactivated SAs** algorithm type (C1-ACCCDCSAR) is provided in the base package. This algorithm type can be used to create an audit algorithm on the Account maintenance object to determine if a service agreement sync request record is to be created for each of the account's non-closed, non-canceled service agreement(s). A change in a related account's details will instantiate a Service Agreement Extract Sync Request (of the type defined for the SA Sync Request BO algorithm parameter) if one does not already exist in the initial state for the service agreement. The base package also contains an algorithm (C1-ACCEDCSA) that is based on this algorithm type.
- A **Person Change Data Capture (SA-Based) – Include Reactivated SAs** algorithm type (C1-ACCCDCSAR) is provided in the base package. This algorithm type can be used to create an audit algorithm on the Person maintenance object to determine if a service agreement sync request record is to be created for each of the person's non-closed, non-canceled service agreement(s). A change in the main person details linked to the service agreement's account will instantiate a Service Agreement Extract Sync Request (of the type defined for the SA Sync Request BO algorithm parameter) if one does not already exist in the initial state for the service agreement. The base package also contains an algorithm (C1-PERECDCSA) that is based on this algorithm type.
- A **Premise Change Data Capture (SP-Based)** algorithm type (C1-PREMCDCSP) is provided in the base package. This algorithm type can be used to create an audit algorithm on the Premise maintenance object to determine if a service point sync request record is to be created for each of the premise's service point. A change in the premise details will instantiate a Service Point Extract Sync Request (of the type defined for the SP Sync Request BO algorithm parameter) if one does not already exist in the initial state for the service point. The base package also contains an algorithm (C1-PREECDCSP) that is based on this algorithm type.
- A **Meter Change Data Capture (SP/Meter History-Based)** algorithm type (C1-MTRCDCSPH) is provided in the base package. This algorithm type can be used to create an audit algorithm on the Meter maintenance object to determine if a SP / Meter History sync request record is to be created for the meter's current installation. A change in the meter details will instantiate a SP / Meter History Extract Sync Request (of the type defined for the SP/Meter History Sync BO algorithm parameter) if one does not already exist in the initial state for the current meter installation. The base package also contains an algorithm (C1-MTRCDCSPH) that is based on this algorithm type.

Master Data Extracts: Business Objects and Algorithms

The maintenance object audit algorithms create data synchronization requests in the following manner:

- If the Generic Change Data Capture (F1-GCHG-CDCP) algorithm type (provided in the base package) is used, the synchronization requests are based on the Sync Request BO maintenance object options.
- Alternatively, if the base package supplied Maintenance Object – Audit algorithm types, which are outlined in the [Master Data Extracts: Maintenance Object Audit Algorithms](#) section are used, the synchronization requests are based on the Sync Request BO defined in the algorithm parameters.

Extraction of service agreements, service points and SP/Meter History related information are supported by the following data synchronization business objects.

- **SA/Account/Person Sync for DataConnect (C1-ExternalRepoSASync):** used to extract service agreement / account / person related information. This business object should be defined as a value for the Sync Request BO option on the Service Agreement maintenance object.
- **SP/Premise Sync for DataConnect (C1-ExternalRepoSPSync):** used to extract service point/premise related information. This business object should be defined as a value for the Sync Request BO option on the Service Point maintenance object.
- **SP/Meter History for DataConnect (C1-ExternalRepoSPMtrHistSync):** used to extract SP/meter history related information. This business object should be defined as a value for the Sync Request BO option on the SP/meter history maintenance object.

These business objects use the following Pre-Processing algorithms to take initial data snapshots, and define the batch control used to extract data and export to a flat file:

- Capture SA-Based Initial Snapshot for DataConnect (C1-CEXTRCSAI)
- Capture SP-Based Initial Snapshot for DataConnect (C1-CEXTRCSPI)
- Capture SP/Meter History Initial Snapshot for DataConnect (C1-CEXTRCSHI)

The Sync Request Monitor batch control (F1-SYNRQ) monitors synchronization requests in the Pending state and executes Monitor algorithms that check for other pending related synchronization requests, and, if none, transitions them to the Determine if Sync Needed state.

Enter algorithms on the Determine If Sync Needed states verify if a synchronization is needed by capturing the final snapshot of the data and comparing it against the initial snapshot. If changes are detected, the final snapshots of the data are stamped on the synchronization requests for export.

- Capture SA-Based Final Snapshot for DataConnect (C1-CEXTRCSAF)
- Capture SP-Based Final Snapshot for DataConnect (C1-CEXTRCSPF)
- Capture SP/Meter History Final Snapshot for DataConnect (C1-CEXTRCSHF)

The Prepare Delimited Extract Data (Ignore Custom Sync Records) Enter algorithm (C1-PRPEXDNCU) on the Send Request state prepares and formats the data for extraction, and creates a general process record for the synchronization request (based on the batch control defined by the pre-processing algorithm).

Master Data Extracts: Batch Controls

The master data extraction process uses a set of batch processes to create data synchronization requests and extract files.

The following batch processes are used to create initial synchronization requests:

- **SA-Based Initial Load for DataConnect (C1-SAEIL):** This batch control creates initial synchronization requests for service agreements / accounts / persons.

- **SP-Based Initial Load for DataConnect (C1-SPEIL):** This batch control creates initial synchronization requests for service points / premises.
- **SP/Meter Initial Load for DataConnect (C1-SMEIL):** This batch control creates initial synchronization requests for SP/Meter History.

The following batch processes are used to create extract files from synchronization requests.

NOTE: These batch processes should be run to create initial load data synchronization requests based on the Sync for DataConnect business objects.

- **SA-Based Extract for DataConnect (C1-SASYX):** This batch control creates extract file(s) for service agreements / accounts / persons related information.
- **SP-Based Extract for DataConnect (C1-SPSYX):** This batch control creates extract file(s) for service points / premises related information.
- **Meter History Extract for DataConnect (C1-SMSYX):** This batch control creates extract file(s) for SP/Meter History related information

These batch controls are defined as values for the Batch Control for Extract algorithm parameters on the Pre-Processing algorithms on the Sync for DataConnect business objects (see above).

Use the Batch Control portal for more information about these batch controls. The extract batch controls contain parameters that can be used to specify details (including path and file name for this file.) for a delimited flat file containing extracted data.

Setting Up and Executing Master Data Extracts

Setting up master data extracts involves the following steps:

1. Add audit algorithms on the Service Agreement, Account, Person, Service Point, Premise, SP/Meter History and Meter maintenance objects.
 - a) Add the Generic Change Data Capture algorithm as an audit algorithm on the Service Agreement, Service Point, and SP/Meter History maintenance objects.
 - b) Add the Account Change Data Capture (SA-Based) - Include Reactivated SAs algorithm as an audit algorithm on the Account maintenance object.
 - c) Add the Person Change Data Capture (SA-Based) - Include Reactivated SAs algorithm as an audit algorithm on the Person maintenance object.
 - d) Add the Premise Change Data Capture (SA-Based) algorithm as an audit algorithm on the Premise maintenance object.
 - e) Add the Meter Change Data Capture (SP/Meter History-Based) algorithm as an audit algorithm on the Meter maintenance object.
2. Add the DataConnect synchronization request business objects as Sync Request BO options on the Service Agreement, Service Point and SP/Meter History maintenance objects.
3. Execute initial load batch processes related to service agreements and service points.
4. Once the initial load synchronization has been executed, changes to service agreements, accounts, persons, service points, premises, SP/Meter History records or meters will trigger the creation of new synchronization requests and resulting extraction files.
5. Execute the sync request monitor process.
6. Execute the extract batch processes related to service agreements and service points.

Billing Data Extracts

When sending billing related data to an external system, both historical and current data needs to be extracted.

Historical data can be extracted as part of an initial load process, and only needs to be provided during initial setup of the integration. Historical data includes data history for all bill segments for a specified historical period that are:

- Linked to completed bills.
- Not linked to bills but are linked to frozen FTs that are linked to completed bills.

Current data should be extracted on a regular ongoing (or incremental) basis. However, in addition to sending current data, any billing corrections should be extracted as well.

Extraction of billing data is performed through use of data synchronization, post bill completion algorithms, business objects and related batch processes.

Billing Data Extracts: Business Objects and Algorithms

Billing generates synchronization requests when bills complete through a Customer Class – Post Bill Completion algorithm. The Change Data Capture of Billing Data for DataConnect (C1-CAPBILLEX) algorithm type provided in the base package can be used to create a Customer Class – Post Bill Completion algorithm that creates a billing synchronization request record for each of the bill's bill segments if one does not exist in the initial state.

Billing related information extracts are supported by the following data synchronization business object.

- Billing Sync for DataConnect (C1-ExternalRepoBillingSync). This business object should be defined as an algorithm parameter on the algorithm based on the Change Data Capture of Billing Data for DataConnect (C1-CAPBILLEX) algorithm type.

This business object use the following Pre-Processing algorithm to take an initial data snapshot, and define the batch control used to extract data and export to a flat file. However, since billing data are always synched when it gets completed, no actual initial snapshot is taken. Instead, the parameters used for building the snapshots are taken and stored on the sync request record for later use when building the final snapshot.

- Capture Billing Data Initial Snapshot for DataConnect (C1-EXTRCBLIP)

The Sync Request Monitor batch control (F1-SYNRQ) monitors synchronization requests in the Pending state and executes Monitor algorithms that check for pending related synchronization requests, and, if none, transitions them to the Determine if Sync Needed state.

Enter algorithms on the Determine If Sync Needed states extract a final snapshot of the data if the billing related data is to be extracted and exported.

- Determine if the Billing Sync is Needed (C1-DSCANUSNS) - this is the algorithm that checks if there is a need to export the billing data. It will discard the synchronization request if one of the following is true:
 - The synchronization request is for a cancelled bill segment whose original usage has not been previously sent out.
 - The synchronization request has been marked for non-synchronization (such as, the bill segments related SA Type is not configured to be extracted on the Billing Data Configuration for DataConnect master configuration record).
- Capture Billing Data Final Snapshot for DataConnect (C1-CAPBILUSN) - this is the algorithm that captures the billing data information and stores it on the synchronization request.

If a single extract file that combines both the billing and master data information is preferred, the following algorithms can be used to capture the master data information of the bill. These algorithms should be plugged in on the Determine if Sync Needed state after the Capture Billing Data Final Snapshot for DataConnect algorithm.

- Capture Billing's SA Final Snapshot for DataConnect (C1-CEXTRCSA) - This is the algorithm that captures the service agreement, account and person information associated with the bill.

- Capture Billing's SP Final Snapshot for DataConnect (C1-CEXTRCSP) - This is the algorithm that captures the service point and premise information associated with the bill.
- Capture Billing's Meter History Final Snapshot for DataConnect (C1-CEXTRCSM) - This is the algorithm that captures the meter history associated with the bill.

The Prepare Delimited Extract Data (Process All) Enter algorithm (C1-PRPEXDCUS) on the Send Request state prepares and formats the data for extraction, and creates a general process record for the synchronization request (based on the extract batch control defined by the pre-processing algorithm).

Billing Data Extracts: Batch Controls

The billing data extraction process uses a set of batch processes to create data synchronization requests and extract files.

The following batch process is used to create initial synchronization requests:

- Billing Data Initial Load for DataConnect (C1-BLEIL): This batch control creates initial synchronization requests for bills.

This batch processes should be run to create initial load data synchronization requests based on the C1-ExternalRepoBillingSync business object.

The following batch process is used to create extract files from synchronization requests.

- Billing Data Extract for DataConnect (C1-BSYEX): This batch control creates extract file(s) for billing related information.

This batch control is defined as a value for the Batch Control for Extract algorithm parameter on the Pre-Processing algorithm on the C1-ExternalRepoBillingSync business object.

Use the Batch Control portal for more information about these batch controls. The extract batch controls contain parameters that can be used to specify details (including path and file name for this file) for a delimited flat file containing extracted data.

Setting Up and Executing Billing Data Extracts

Setting up billing data extracts involves the following steps:

1. Configure the Billing Data Configuration for the DataConnect master configuration record. This is used to configure the UOM, TOU, and SQI of the SA Type's service quantity and charges that will be extracted. Additionally, you can also configure the UOM, TOU and SQI at the rate level.
2. Add the Change Data Capture of Billing Data for DataConnect (C1-CAPBILLEX) related post bill completion algorithm to the applicable customer classes.
3. Execute initial load batch processes related to billing data.
4. Once the initial load synchronization has been executed, current data should be extracted on a regular ongoing (or incremental) basis. In addition to sending current data, any billing corrections will be extracted as well (data related to bill segments not linked to bills but are linked to frozen FTs that are linked to completed bills).
5. Execute the sync request monitor process.
6. Execute the extract batch processes related to billing.

Billing Data Configuration for DataConnect Master Configuration

The master configuration business object for Billing Data Configuration for DataConnect defines general configuration details for the DataConnect functionality. This is used to configure the UOM, TOU, and SQI of the SA Type's service quantity and charges that will be extracted. Additionally, you can also configure the UOM, TOU and SQI at the rate level.

Navigate using **Admin menu > Master Configuration**. In the Master Configuration list, scroll to Billing Data Configuration for DataConnect and click the add icon if there is no record yet or the edit icon to modify an existing record.

For more information about specific fields in the master configuration, refer to the embedded help.

Extract Flat File Formats

This section outlines the format of output flat files created by master and billing data extracts.

Data Areas:

- [C1-ExternalRepoSABasedSnapshot](#)
- [C1-ExternalRepoSPBasedSnapshot](#)
- [C1-ExternalRepoSPMtrHtSnapshot](#)
- [C1-ExternalRepoBillingSnapshot](#)

C1-ExternalRepoSABasedSnapshot

C1-ExternalRepoSABasedSnapshot is used for the SA-based extracts and includes the following information:

Field Name	Format	Maximum Size	Comments
Service Agreement Id	String	10	<i>Required</i>
Account Id	String	10	<i>Required</i>
Person Id	String	10	<i>Required</i>
Bill Cycle	String	4	
Bill Cycle Description	String	60	
Collection Class	String	10	
Collection Class Description	String	60	
Account Information	String	254	<i>Required</i> The information string constructed for the account by the Account Information algorithm on the Installation Options
Customer Class	String	8	<i>Required</i>
Customer Class Description	String	60	<i>Required</i>
Landlord Indicator	String	1	<i>Required</i> If the account on the service agreement is the same as the account for the landlord on the service agreement's premise, then this is set to 'Y'; otherwise it is set to 'N'

Mailing Address	String	254	
Mailing Address 2	String	254	
Mailing Address 3	String	254	
Mailing City	String	90	
Mailing State	String	6	
Mailing Postal	String	12	
First Name	String	254	
Last Name	String	254	<i>Required</i> If the customer is a business, the business name is populated here
Phone Number 1	String	254	
Phone Number 2	String	254	
Email Address	String	254	
Special Role Flag	String	2	
Special Role Description	String	60	
Service Type	String	2	<i>Required</i>
Service Type Description	String	60	<i>Required</i>
CIS Division	String	5	<i>Required</i>
CIS Division Description	String	60	<i>Required</i>
SA Type	String	8	<i>Required</i>
SA Type Description	String	60	<i>Required</i>
Revenue Class	String	8	<i>Required</i>
Revenue Class Description	String	60	<i>Required</i>
SIC Code	String	8	
SIC Code Description	String	60	
Deposit Class	String	8	
Deposit Class Description	String	60	
Campaign	String	12	
Campaign Description	String	60	
Debt Class	String	4	<i>Required</i>
Debt Class Description	String	60	<i>Required</i>
SA Start Date	Date	10	<i>Required</i>
SA End Date	Date	10	<i>Required</i>
SA Status	String	2	<i>Required</i>
SA Status Description	String	60	<i>Required</i>
Rate Schedule	String	8	<i>Required</i>
Rate Schedule Description	String	60	<i>Required</i>
SA's Service Point(s) Id	String	10	
SA's Service Point(s) Start Date/ Time	Date/Time	26	

[Top of Page](#)

C1-ExternalRepoSPBasedSnapshot

C1-ExternalRepoSPBasedSnapshot is used for the SP-based extracts and includes the following:

Field Name	Format	Maximum Size	Comments
Service Point Id	String	10	<i>Required</i>
Premise Id	String	10	<i>Required</i>
Address Line 1	String	254	<i>Required</i>
City	String	90	
State	String	6	
Postal	String	12	
County	String	90	
Country	String	3	
Premise Type	String	8	<i>Required</i>
Premise Type Description	String	60	<i>Required</i>
CIS Division	String	5	
CIS Division Description	String	60	
Life Support Sensitive Load	String	2	
Life Support Sensitive Load Description	String	1000	
Trend Area	String	8	<i>Required</i>
Trend Area Description	String	60	<i>Required</i>
In City Limit	String	1	<i>Required</i>
Address Line 2	String	254	
Address Line 3	String	254	
Number 1	String	6	
Number 2	String	64	
Landlord Id	String	10	
Landlord Description	String	60	
SP Status	String	2	<i>Required</i>
SP Status Description	String	60	<i>Required</i>
Service Cycle	String	16	<i>Required</i>
Service Cycle Description	String	60	<i>Required</i>
Service Route	String	16	
Service Route Description	String	60	
Service Route Sequence	Number	8	
Meter Location	String	4	

Meter Location Description	String	60	
Latitude	Number	2.7	
Longitude	Number	3.7	
Facility Level 1	String	8	
Facility Level 2	String	8	
Facility Level 3	String	8	
SP Disconnect Location	String	4	
SP Disconnect Location Description	String	60	
SP Source Status	String	2	<i>Required</i>
SP Source Status Description	String	60	<i>Required</i>

[Top of Page](#)

C1-ExternalRepoSPMtrHtSnapshot

C1-ExternalRepoSPMtrHtSnapshot is used for the SP/Meter History based extracts and includes the following:

Field Name	Format	Maximum Size	Comments
Service Point Id	String	10	<i>Required</i>
Meter Type	String	8	<i>Required</i>
Meter Type Description	String	60	<i>Required</i>
Meter Status	String	2	<i>Required</i>
Meter Status Description	String	60	<i>Required</i>
Register Constant	Number	6.6	<i>Required</i>
Installation Constant	Number	6.6	<i>Required</i>
Meter's Number of Dials	Number	2.2	<i>Required</i>
Manufacturer	String	8	<i>Required</i>
Manufacturer Description	String	60	<i>Required</i>
Model	String	8	<i>Required</i>
Model Description	String	60	<i>Required</i>
Badge Number	String	60	<i>Required</i>
Serial Number	String	16	
Meter Id	String	10	<i>Required</i>
Installation Date/Time	Date/Time	26	<i>Required</i>
Removal Date/Time	Date/Time	26	
SP Meter On/Off Status	String	2	<i>Required</i>

[Top of Page](#)

C1-ExternalRepoBillingSnapshot

C1-ExternalRepoBillingSnapshot is used for the Billing-based extracts and includes the following:

Field Name	Format	Maximum Size	Comments
Billable Service Quantity	Number	12.6	<i>Required</i> Usage value billed to the customer during this period
Bill Segment End Date	Date	10	<i>Required</i> End date of the bill period
Consumption Days	Number	2	<i>Required</i> Number of days the billable service quantity represent
Bill Segment Estimate Indicator	String	1	<i>Required</i> A = actual reading ; E = estimated reading
Calculated Amount	Number	13.2	Amount billed to the customer for usage during this period
Unit of Measure	String	4	<i>Required</i>
Time of Use	String	8	
SQI	String	8	
Service Agreement ID	String	10	<i>Required</i>
Service Point ID	String	10	<i>Required</i>

[Top of Page](#)

External Fieldwork System Integration

The following section describes functionality provided for integrating your field activities with an external system.

NOTE:

Separate module. Field activity integration functionality is associated with separate Field Activity Integration module. If this module is not applicable to your business you may turn it off. Refer to [Turn Off A Function Module](#) for more information.

The Big Picture of External System Integration

Many utilities use other systems to coordinate work that goes out to the field. The following are examples of functionality provided by fieldwork management systems:

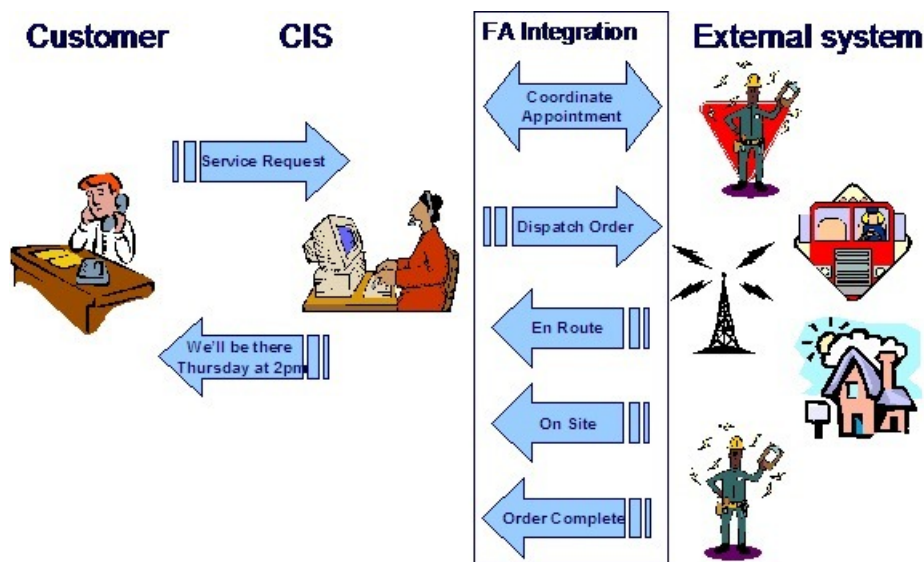
- Defining crews, skills, availability for work
- Scheduling appointments
- Tracking individual field activities, usually utilizing handheld devices to get up-to-the-minute status
- Balancing the workload

Since Oracle Utilities Customer Care and Billing can generate many work orders, you can integrate it with your external systems. The most common type of external system used to coordinate fieldwork is a Workforce Management (WFM)

system. However, some companies may set up their business such that certain field activities should be sent to other types of systems. For example,

- Maybe new meter installations are the responsibility of the Asset Management system. A field activity of this type originated in Oracle Utilities Customer Care and Billing should be sent to the asset management system. That type of system may support its own crew dispatching logic or perhaps it performs some simple processing for the field activity and then interfaces the field activity to the WFM system.
- Outage related field activities are typically sent to an outage management system. The outage management system uses information from several outage calls to determine the source of the outage problem. The outage system may support its own crew dispatching logic or it may integrate with a WFM system to dispatch a crew.

The following diagram illustrates the typical integration with an external system:



The following points describe the diagram:

- While speaking with the customer and entering information into the system, information about available appointments may be accessed from an external system and displayed on an Oracle Utilities Customer Care and Billing user interface. Information about the appointment that is booked with the customer is sent to the external system.
- Information about new field activities created in Oracle Utilities Customer Care and Billing is sent to the appropriate external system using XAI. The external system performs the dispatching and tracking for the field activity.
- Intermediate field activity states may be interfaced from the external system to Oracle Utilities Customer Care and Billing.
- Completion information is interfaced from the external system to Oracle Utilities Customer Care and Billing.

Mapping Field Activity Types to an External System

When integrating with an external system, you must determine how to map your field activity types in Oracle Utilities Customer Care and Billing to your related task identifier in the external system.

Decide whether Oracle Utilities Customer Care and Billing provides information on its field activity types that map to the task in the external system. For more information, see [Designing Your Field Activity Profiles & Types](#). You can leverage Oracle Utilities Customer Care and Billing field activity types and the messages sent from Oracle Utilities Customer Care and Billing to indicate the appropriate task in the external system. If tasks in the external system already have the information that maps to the field activity types in Oracle Utilities Customer Care and Billing, then the messages in Oracle

Utilities Customer Care and Billing can pass the field activity types to the external system. You need to determine where your mappings will exist to decide how to map the field activity types and where.

In the sample integrations provided, it is assumed that the Oracle Utilities Customer Care and Billing field activity types map to the external system tasks using characteristics on the field activity type. You can use one or more characteristics on the field activity type to map it to the task in the external system.

Canceling a Field Activity

Various business processes in the system create field activities (for example, start/stop, severance or cut process, device test selection). If the process that creates a field activity is canceled, the system attempts to automatically cancel the related field activity. If an implementation integrates with an external system, typically messages can be sent real-time to the external system AND to a field worker dispatched to work on the field activity to indicate that an FA is being canceled.

However, it's possible that there are situations where it is known ahead of time that a message cannot be successfully sent real-time. (For example, some field workers do not have real time communication with the dispatcher while in the field).

If this situation exists, you may configure the [external system feature configuration](#) to identify intermediate status values that should prevent a field activity from automatically completing. All processes that attempt to automatically cancel a field activity whose dispatch group references an external system will first check to see if it's in an intermediate status that should prevent auto-cancellation.

NOTE:

Each business process must define what should occur when a field activity is not allowed to be auto-canceled.

Field Activities Not Related to a Premise

There are often field activities generated that do not relate to a specific premise. Because Oracle Utilities Customer Care and Billing requires a service point for every field activity, implementations must define one or more premises to represent the "no address premise" to use for one of these field activities. The decision as to whether to use one premise or more than one premise depends on how you plan to interface this information to an external system. For example

- Maybe the external system expects the user to use the field activity instructions to describe the location of the problem. In this case only one "no address" premise is needed
- You may decide that the operations area on a field activity's service point is interfaced to the external system and that you want to create a different "no address" premise for each of your operations areas.

Regardless of how many "no address" premises you decide to define, you must create at least one service point to be able to create a field activity. You may choose to create a service point for each service type you support. You must also be sure that your users know how to find the correct premise and service point in the search when creating the field activity.

We recommend that you create a special [premise type](#) to use for these premises and reference that premise type in the No Address Premise Type option on the [feature configuration](#). The base product [Extract FA Information](#) service resets the Address fields if the premise linked to the field activity's service point is a "no address" premise type.

Override Phone Number

When entering field activity information for a customer, you may want to capture a specific contact phone number to interface to the external system. We recommend that you create a field activity characteristic type to capture the contact phone number and reference that characteristic type in the Phone Number Characteristic Type option on the [feature configuration](#). The base product [Extract FA Information](#) service overrides the Account's phone number if it finds a characteristic on the field activity with this characteristic type.

Integration Through XAI

The integration between Oracle Utilities Customer Care and Billing and the various external systems is through XAI.

- The XAI [real time outgoing message](#) engine is used to retrieve appointment information and book appointments.
- Information about field activities is interfaced to the external system using [near real time NDS messages](#).
- When the system receives messages from the external system, records are received as [inbound messages](#).

External System Feature Configuration Refers to a Service Provider

In order to communicate to the external system using XAI, the system must be able to identify a service provider related to the external system. The service provider associated with an external system is defined as an option on the [external system's feature configuration](#).

Dispatching Field Activities

When your implementation integrates with an external system, field activities are created in Oracle Utilities Customer Care and Billing and interfaced to the external system. The external system manages the assigning and dispatching of work associated with this field activity.

If your external system can supply interim status information, this can be passed to our system and logged with the field activity. For example, if the external system can indicate that the field worker is en route or on site, this can be logged for the field activity so that if the customer calls to inquire about the status, the user is able to communicate the information.

When work is completed, full completion information related to the field activities is interfaced from the external system back to the system.

NOTE:

that field activities are interfaced to the external system, not field orders. Field orders are not necessary because they are used to bundle field activities together, an activity now managed by the external system. You may still choose to create field orders for these types of field activities if you want to group field activities for your own purposes.

Integration with Multiple External Systems

Some organizations use more than one external system to manage different types of field activities. For example, line work may be handled in one system and meter service work in another. To achieve this the [external system's feature configuration](#) is defined at the dispatch group level. Field activities are interfaced to an external system based on their dispatch group information.

Algorithms Control FA Integration

The interaction between Oracle Utilities Customer Care and Billing and your external systems is handled entirely via algorithms that are plugged into the appropriate dispatch group. If your dispatch group indicates that it should interface with an external system, then you must specify one or more FA Integration algorithms. These algorithms are called when the following system events occur

- A field activity linked to the dispatch group is created, whether online or in batch
- A field activity linked to the dispatch group is canceled, whether online or in batch
- The "header" information for a field activity linked to the dispatch group is changed, whether online or in batch

NOTE:

Header information. "Header" information for a field activity refers to information on the field activity record. It does NOT include steps, characteristics, remarks or log records.

- The characteristics collection for a field activity changes online. Changing the characteristics via a batch routine will not trigger the FA integration algorithms.

Sample FA Integration Algorithm

The system provides a sample algorithm for FA integration called [FWFM-FA-INT](#). This algorithm creates XAI outbound messages (by creating NDS records) to notify an external system if any field activities are created or if any changes are made to existing FAs so that their corresponding orders are created or updated respectively. Refer to the algorithm type description for more detail about the algorithm's logic.

NOTE:

Alert For Problems With Message. The base product provides an alert algorithm ([CCAL-FAERMSG](#)) that may be plugged in to the [installation options](#). This algorithm displays an alert if there is an error associated with an outbound message for the field activity or if a response has not been received within a specified time limit.

Extract FA Information

When XAI processes the outgoing messages generated by the sample FA integration algorithm, it builds the XML request by calling the service referenced on the NDS type's XAI inbound service.

The base product provides a service, called CIPOEFIP, which extracts field activity information along with information related to the service point, meter, item, premise and account for the field activity (if applicable). If this service does not provide the functionality required by your implementation, you may create your own version using this one as a basis.

Note that this service defines an input parameter to exclude Financial Information. You can set this to Yes to avoid the account financial information (see below).

The following tables list the data that is extracted.

FA Type information

All fields in the main FA Type table and the corresponding language table

FA Type Characteristics

FA Information

All fields in the main Field Activity table

Status description

Cancellation Reason description

FA Steps collection

FA Characteristics. Note the service limits the number of field activity characteristics that it extracts. It only extracts the first 100.

Appointment information

SP Type information

All fields in the main SP Type table and the corresponding language table

SP Type Characteristics

SP Information

All fields in the main Service Point table

Status description

Source Status description

Disconnect Location description

SP Location description

Facility Level 1, 2 and 3 descriptions

SP Operation Area and Description for the FA type's field service class

SP Multi-Item collection

SP Equipment collection

SP Characteristics

SP Geographic Data

SP Operation Area collection

Premise Information

All fields in the main Premise table

Premise Type description

MR Warning description

MR Instructions description

Life Support/Sensitive Load flag description

Alternate Address collection

Landlord Agreement Description

Name of Landlord's main person

Phone Number collection of Landlord's main person

Premise Characteristics

Premise Geographic Data

Meter Information for the meter currently installed at the Service Point (if applicable)

All fields in the main Meter table

Meter Type description

Meter Status description

Manufacturer description

Model description

Meter Id collection

Meter Characteristics

Meter Equipment collection

Meter Configuration Information for the meter currently installed at the Service Point (if applicable)

All fields in the main Meter Configuration table

Meter Installation Date/Time

Calculated High / Low limit for the meter on the FA schedule date

Item Information for the badged item currently installed at the Service Point (if applicable)

All fields in the main Item table

Item Type description

Item Status description

Manufacturer description

Model description

Item Characteristics

Item Equipment collection

SA Information for the first two Service Agreements found for the service point

SA ID

CIS Division / SA Type

Account ID

SA Status

Start Date / End Date

Customer Read Flag

Rate Schedule

SA / SP collection

Customer Information for the account(s) linked to the above service agreements

All fields in the main Account table

Person ID of the main person

Name of the main person

Person / Business flag for the main person

Life Support / Sensitive Load flag, flag Description and Description for the main person

Phone Number collection for main person. Note, this information may be [overridden](#) if a phone number is entered on the field activity.

The following financial information is not extracted if the input Exclude Financial Information is set to Yes.

Credit Rating Points. Calculated from the Credit Rating Base Score on the Installation Table and CR_RATING_PTS from CI_ACCT_CR_R_VW

Cash Only Points. Calculated from the Cash Only Base Score on the Installation Table and CASH_ONLY_PTS from CI_ACCT_CR_R_VW

Current Amount (calculated from FT information)

Payoff Amount (calculated from FT information)

Disputed Amount (calculated from FT information)

Arrears Information: Disputed Amount, New Charges, 30 days old, 60 days old, 90 days old

Count of the number of bills generated in the last 6 Months

For each bill:

Bill ID

Bill Date

Bill Amount

Meter Read information for the last Bill

All fields in the main Meter Read table

Register Read collection

Count of the number of payments generated in the last 6 Months

For each payment:

Payment ID

Payment Event ID

Payment Date

Payment Amount

Cancel Reason and its Description

Incoming Messages from the External System

If your implementation requires general updates to the field activity from your external system, for example, updating the priority based on a change to the priority in the external system, the XAI Inbound Service that processes this message may reference the standard field activity service.

The base product supplies two additional services to handle special incoming messages from your external system.

- One service processes responses to outgoing messages initiated by Oracle Utilities Customer Care and Billing.
- One service processes field activity completion information.

FA Response

The sample integration is designed to expect a response, whether positive or negative, from the external system when an outgoing message is sent.

The base product provides a service, called CIPORFAP, which processes this response. Note that we also provide an XAI Inbound Service, FA Response, defined to invoke this service. This service provides the ability to do the following for positive responses:

- Update information about the field activity (refer to [Populating FA Response](#) for details)
- Create a [field activity log](#) entry

If the incoming message is a negative response to a message originally sent by Oracle Utilities Customer Care and Billing, the service supports the following functionality:

- Create a [field activity log](#) entry.
- Create a To Do Entry

For both the log entry and the To Do entry, if the external system has provided an external message category and code in the message that have been mapped to an internal message category and code on the feature configuration table, the system looks up the message text and includes it in the log message. Refer to [FA Log Entry Events](#) for more information.

The following topics provide some additional information.

Create a Log Entry

In order to create a log entry, the system needs to know the log type to use. The service accepts a log type as input and your XSL that maps data to this service may assign an appropriate log type.

If a log type is not provided, the system can try to default one of the base sample log types if information is provided about the type of base sample message that this is a response to. To do this, the service needs to identify the notification download condition of the original message.

FASTPATH:

Refer to [Sample FA Integration Algorithm](#) for information about the notification download conditions used for the sample integration.

Your XSL may provide

- The notification download condition flag.
- The notification download type. Using the NDS type, the system can find the appropriate notification download condition flag.
- The external system and the message id that the system generated and sent to the external system. Using that information, the original outbound (NDS) message can be retrieved and the NDS type and its corresponding notification download condition flag can be determined.

FASTPATH:

Refer to [Sample FA Integration Algorithm](#) for more information about creating the message ID.

Refer to [Field Activity Log](#) for a list of the base sample log types and the type of message they are generated for.

Create a To Do Entry

In order to create a To Do entry, the system needs to know the To Do type to use. The service accepts a To Do type as input and your XSL that maps data to this service may assign an appropriate To Do type. If you don't pass in a To Do type the system looks up the **To Do Type for FA Response** indicated on your [external system's feature configuration](#).

Populating FA Response

The following table lists the fields to populate for an FA response message.

General information

FA ID

Positive Switch (Y if response is positive, N if response is negative)

Outbound Message Information (used to identify the original outgoing message that this is a response to)

External System

Message ID

Notification Download Condition flag

NDS Type

Error Message Information Refer to [FA Response](#) for more information

Message Category

Message Number

FA Log Information

Create Log Switch (Y, N)

FA Log Type (Refer to [Create a Log Entry](#) for more information)

To Do Information

Create To Do Switch (Y, N)

To Do Type (Refer to [Create a To Do Entry](#) for more information)

FA Information

Schedule Date/Time

Dispatch Group

Instructions

Comments

External ID

FA Intermediate Status

Appointment Information

Field Activity Characteristics

Intermediate Status Updates

Oracle Utilities Customer Care and Billing supports receipt of intermediate status updates for a field activity from an external system. To process messages from the external system regarding status update changes, the system call the standard Field Activity service with a change action updating the intermediate status as provided.

If your organization's external system supports different intermediate status values for a [field activity](#), your implementers must customize the lookup value that defines the list of valid intermediate status values.

Field Activity Completion

The system provides a staging table to use to upload and complete field activities. However, this staging table has limitations as described in [field activity completion considerations](#). For these reasons and to enable to support [integration through XAI](#), the base product provides a sample service (called CIPOASTP) that can be invoked by XAI to support completion of the all step types, including the "generic" step types and the "standard" step types. We provide an XAI Inbound Service, C1FACompletionWithSteps, defined to invoke this service. The sample service:

- Creates field activity upload staging and FA step upload staging records
- If meter read information has been provided, it creates a meter read record and its corresponding register reads.
- For any of the "generic" steps, the process updates or creates the appropriate record(s) and populates the id of the record updated as the foreign key for the FA step record. For example, for the Change Meter step type, the process updates the meter record with the information provided through XAI and links the meter id to the FA step record.
- The process then performs the standard "complete step" logic. For the "generic" step types, the FA step should simply be marked as complete because the foreign key is already linked. For the "standard" step types, the completion logic described in [FACOMPL - Upload and Complete Field Activities](#) is performed.

FASTPATH:

Refer to [Field Activity Completion Considerations](#) for the list of "generic" and "standard" step types.

- Creates an FA log entry with a log type of Order Completion to indicate that the field activity has been completed by an external system.
- Standard FA completion logic is also performed. For example, completion algorithms are executed, etc.

NOTE:

The sample service provided by the system may not provide all the functionality your implementation requires for completing every "generic" step. For example, not all service point, meter or item fields are included in the list of fields that may be updated. If the sample process does not satisfy your needs, your implementers should copy the sample process and modify the new process as needed.

Characteristic limitation. The sample service limits the number of field activity characteristics that may be uploaded to 100.

Error Handling. If any error is found during step completion, this sample service backs out all changes and issues an error indicating the problem.

Populating Field Activity Completion

The following table lists the fields to populate for a Field Activity Completion.

FA Upload Staging Information
FA ID
All fields in the FA Upload Staging table, FA upload characteristics, FA upload remarks and FA upload staging steps
"Generic" Step Type Information
Step Sequence Number
Customer Contact Info
Customer Contact Date / Time
Customer Contact Class
Customer Contact Type
Customer Contact Comments
Meter Information
Meter Type
Meter Status
Manufacturer
Model
Serial Number
Receive Date
Retirement Date
Comments
Retire Reason Code
Meter ID Collection
Meter Characteristics
Meter Configuration Information
Effective Date / Time

Meter Configuration Type

Register Collection

Item Information

Item Type

Item Status

Manufacturer

Model

Serial Number

Receive Date

Retirement Date

Comments

Retire Reason Code

Item Characteristics

SP Information

SP Type

SP Status

Installation Date

Abolish Date

SP Source Status

Disconnect Location Code

Service cycle

Service Route

Service cycle / Route Sequence

Meter Location Code

Meter Location Details

Comments

SP Characteristics

SP Multi-Item Information

Effective Date for new collection

Count of Items

Item Type Code

Item Count

Item Difference. Used to indicate only changes to the existing item count collection.

Device Test Information

All fields for main Device Test table

External System. Used to record a test done by a third party.

Device Test Component / Result Extra Information.

This additional information is used to identify the meter readings that should be associated with component test results that are meter readings.

Note. This FA completion service supports providing information for a single meter reading (in the FA upload staging info above). If your device test produces multiple readings, the assumption is that the readings are entered separately prior to this upload of completion information.

Component sequence. Indicate the sequence of the Device Test Component

Read Sequence for the register for that component. This should correspond to the register's relative sequence within the meter configuration.

Component Result sequence. Indicate the sequence of the Component Result

Read Date/Time. For each result that is related to a meter read, indicate the read date / time so that the system can find the associated meter read for this result.

Booking Appointments Via An External System

If your field activity requires an appointment, the user navigates to the [appointment](#) page to book the appointment. If the dispatch group for your field activity is associated with an [external system's feature configuration](#), the appointment page includes a user exit to provide the ability to communicate with the external system for the following actions:

- Display available appointments
- Book an appointment
- Cancel an appointment

NOTE:

Implementation specific behavior. The actual behavior of your appointment integration is dependent on your external system behavior. For example, some systems store appointment booking, which may be done real time, independently from storing field activities, which are interfaced in near real time. Other systems may display available appointments real time but may book appointments as part of storing the field activity, which is near real time.

This section describes the tools provided to interface with your external system along with a description of sample integration of real time appointment interaction provided with the system.

NOTE:

Oracle Utilities Mobile Workforce Management integration. In addition to general FA integration logic, the system provides integration with Oracle Utilities Mobile Workforce Management. Refer to the documentation for integration for more information.

Configuring Appointment Options

There are several configuration options available to customize the interaction with your external system for appointments.

Manual Appointments

If your external system books appointment in real time, it's possible that there is a problem with the communication to the external system while the user is attempting to book an appointment. You may configure your [external system's feature configuration](#) options to allow manual booking of appointments so that the user is able to book appointments even if the system is down. If you set the option Allow Manual Appointments to Y,

- If the communication is available to show appointments, but the connection is down when you attempt to book one of the displayed appointments, you can click OK to book the appointment using a "near real time" message. This is applicable for systems that book appointments real-time via the appointment page.

- If the communication is unavailable while you are attempting to show appointments, you may add your own appointment period. When you click OK, a "near real time" message is created.

FASTPATH:

Refer to [Book Appointment](#) for more information about the creation of "near real time" messages.

NOTE:

If the user has created an appointment or has chosen an appointment that is not available in the external system, it is possible that the manual appointment is not accepted and a negative response is received. You may configure your system to allow [forced appointments](#) to force the external system to accept the chosen appointment.

If there is a problem with the communication to your external system while attempting to cancel an appointment, you may configure your [external system's feature configuration](#) options to Allow Manual Appointment Cancellation. If this option is set to Y and the communication is unavailable while you are attempting to cancel an appointment, you can click OK to cancel the appointment using a "near real time" message.

Narrowing Appointment Window

If you have set the [external system's feature configuration](#) option Allow Narrowing Of Appointment Window to Y, then your users are able to enter a more granular appointment time than what is displayed. For example, if the list of available appointments shows an appointment period of 1pm to 4pm and this option is turned on, the user could enter an appointment period of 2pm to 2:30pm.

Forced Appointments

If your external system allows the user to choose appointment periods that are not available in the system, set the [external system's feature configuration](#) option Allow Forced Appointments to Y. When this option is turned on a Forced checkbox is visible on the search for an appointment page. The user should check this if the desired appointment is not available.

NOTE:

Forcing manual appointments. If your system allows [manual appointments](#) and allows forced appointments, the [book appointment](#) logic provided with the system automatically flags manual appointments as forced to ensure that they are accepted by the external system.

If an appointment is forced, the system populates a field activity characteristic indicating this. The characteristic type to use is defined as a [feature configuration](#) option.

User Defined Search and Result Fields

It is possible that your external system allows the user to provide additional information prior to searching for available appointments. For example, imagine that your customer wants the appointment to be in the afternoon and your external system allows you to request "afternoon" appointments only. To enable this functionality, you define a User Defined Search Criteria field on the [external system's feature configuration](#) options.

You may define up to 10 user defined search criteria fields. The information is passed to the message engine to [get available appointments](#). It is assumed that your XSL scripts correctly map the information into a format understood by the external system to determine the desired appointment periods.

The result grid for the available appointments displays the start date / time and end date / time. If your external system provides additional information for each appointment period that would help the user choose the best appointment, you may define a User Defined Result Field on the [external system's feature configuration](#) options.

You may define up to 10 user defined result fields. If you have configured the external system to define extra result fields, it is assumed that your response XSL correctly maps the information from the external system to the appropriate column in the available appointments collection so that it can be displayed to the user.

For any custom field that you want to include in the search or the results, you must define a [field](#) in the system to indicate the type of data and the label for this field.

Appointment Periods vs Reservations

When an external system is used for appointments, it is not necessary to set up [appointment period](#) records in Oracle Utilities Customer Care and Billing ahead of time. The external system is responsible for providing the available appointment periods. When a user books an appointment with an external system, Oracle Utilities Customer Care and Billing creates an appointment period as an audit. This enables the user to view the appointment information when viewing the field activity in the system.

If the external system creates a reservation record for the appointment in its system, the unique identifier of that reservation may be stored with the field activity in Oracle Utilities Customer Care and Billing as a characteristic. The characteristic type to use is defined as a [feature configuration](#) option. The sample integration provided with the system populates the field activity characteristic with a reservation number received.

Real Time Appointment Interaction

The appointment page communicates with the external system for appointments via the [XAI real time outgoing message engine](#). This section describes some technical information related to the logic delivered with the system.

Appointment Page User Exit

The base product appointment page provides java user exit code that is invoked when the dispatch group is associated with an external system (i.e., it references a [feature configuration](#)). The user exit code does the following:

- It determines the [service provider](#)
- It finds the NDS types that are associated with the following notification download condition flags:
 - Get Available Appointments. This is passed to the engine to retrieve the available appointments
 - Appointment Book. This is passed to the engine to book an appointment real time.
 - Appointment Book - Near Real Time. This is passed to the engine to book an appointment in near real time.
 - Appointment Cancel. This is passed to the engine to cancel an appointment real time.
 - Appointment Cancel - Near Real Time. This is passed to the engine to cancel an appointment in near real time.
- It invokes the Appointment Java Class Interface defined on the [external system's feature configuration](#) options passing all the data available on the page service.

NOTE:

Implementation specific business logic. The intention is that any unique business logic required to interact with your implementation's external system is encapsulated in the appointment java class interface plugged in on your feature

configuration options. However, if your implementation has unique logic that must be coded in the java user exit on the appointment page, that code may also be replaced by implementation specific appointment page user exit code.

Sample Appointment Java Class

The Appointment Java Class Interface referenced on your external system's feature configuration is responsible for interaction with the XAI real time outbound message engine to communicate with the external system for appointment logic.

The base product provides a default java class (called `com.splwg.wfmi.workforce.DefaultWFMSystem`) for appointment integration that may be used if it provides the logic your implementation needs.

NOTE:

Oracle Utilities Mobile Workforce Management. Refer to documentation on integration for information about the java class provided for integration with Oracle Utilities Mobile Workforce Management.

The default java class provided with the base product does the following:

- Extracts additional field activity information not provided by the appointment page service
- Converts the data in the page service and the additional FA information into an XML document
- Invokes the [real time outgoing message](#) engine.

Responses received from the engine are in the form of an XML document. The java class transforms the information into a format recognized by the page data model and sends it back to the user exit.

NOTE:

The `CILOAPTP.xml` found on the `xmlMetaInfo` directory describes the base structure of the Show Appointment, Book Appointment and Cancel Appointment request XML. The selected field activity to be booked or canceled has an additional `faExtraInfo` element. The `faExtraInfo` element follows the structure described by `CILOEFIP.xml`.

The following sections describe more detail about the logic provided by the sample base product appointment Java class interface.

- [Get Available Appointments](#)
- [Book Appointment](#)
- [Cancel Appointment](#)

Get Available Appointments

For obtaining available appointments, the user exit passes the NDS type that references the Get Available Appointments condition flag. All the data available on the page service is passed to the engine as an XML document including any [user defined search fields](#).

The user exit expects a response to this message to return a collection of records to display in the available appointments grid on the appointment page, including any [user defined result fields](#). Any errors received are communicated to the user.

NOTE:

Translate Message. Any error message received from the external system is translated from an external message to an appropriate system error message using the message information on the [external system's feature configuration](#).

[Top of the Page](#)

Book Appointment

Once the user has confirmed the desired appointment with the customer, the user attempts to book the appointment. The generic appointment integration java class provided with the system sends a message to books appointments real time. The user exit passes the NDS type that references the Appointment Book condition flag.

NOTE:

One FA at a time. The sample user exit provided by the system only supports booking appointments for one field activity at a time. As a result, if you want to use the sample user exit, your external systems should be configured with the option Allow Multiple Reservations set to N. If your organization would like to support booking appointments for multiple field activities at once, you may create your own user exit to provide this capability.

The sample user exit provided with the product expects either a positive or negative response to this message.

- If a positive response is received, the user exit expects to be passed a reservation number and an indication of whether or not the appointment was forced. It populates the field activity characteristics collection with these values using the **Reservation Characteristic Type** and **Appointment Forced Characteristic Type** defined as options on the [external system's feature configuration](#).
 - If a negative response is received, an error message is displayed to the user.
-

NOTE:

Translate Message. Any error message received from the external system is translated from an external message to an appropriate system error message using the message information on the [external system's feature configuration](#).

If the message engine cannot communicate with the external system, it returns an indication to the user exit. The user exit proceeds as follows:

- If the external system indicates that Allow Manual Appointments is set to false an error is displayed to the user.
- If manual appointments are allowed, the user exit issues a warning to the user asking if the message should be logged and sent when the communication is up again. If the user agrees, the user exit invokes the java class asking it to post a [near real time message](#).
 - The user exit calculates a unique outgoing message ID for the external system. The message id is calculated using a database sequence whose name is referenced in the option **Message ID Database Sequence Name** on the [external system's feature configuration](#). This message id is passed to the real time message engine to be populated as an NDS context entry to support an [asynchronous response to the message](#).
 - The user exit passes the NDS type that references the Appointment Book - Near Real Time condition flag. The FA Id is also passed to the real time message engine to be posted as an NDS context entry. The system expects that the response to this message will create an XAI upload staging record and that this record will update the field activity's characteristics with the reservation and forced appointment information (if applicable). Refer to [Near Real Time NDS Messages](#) for more information about responses to near real time messages and XAI upload staging.
 - If the external system's configuration indicates that [forced appointments](#) are allowed, the message is sent to the external system with the forced indication set. If the external system does not allow forced appointments, it's possible that this manual appointment could be rejected by the external system.

If no error is received, the appointment page continues with the "change" action. An appointment period is created for the chosen appointment time if one doesn't already exist and the appointment period is linked to the field activity.

NOTE:

that the appointment period is created and linked to the field activity even if the message is sent to the external system in near real time and no confirmation has been received. This was done to record the requested appointment in our system to cater for the situations when the customer wants to change or cancel the appointment prior to receiving the acknowledgement from the external system.

[Top of the Page](#)

Cancel Appointment

If the customer wants to cancel the appointment, the user navigates to the appointment page. The generic appointment integration java class provided with the system sends a message to cancel an appointment real time. The user exit passes the NDS type that references the Appointment Cancel condition flag. The user exit expects either a positive or negative response to this message.

- If a positive response is received, the appointment page continues with the change action. (See below).
- If a negative response is received, an error message is displayed to the user.

NOTE:

Translate Message. Any error message received from the external system is translated from an external message to an appropriate system error message using the message information on the [external system's feature configuration](#).

If the message engine cannot communicate with the external system, it returns an indication to the user exit. The user exit proceeds as follows:

- If the external system indicates that Allow Manual Appointment Cancellation is set to false an error is displayed to the user.
- If manual appointments are allowed, the user exit issues a warning to the user asking if the message should be logged and sent when the communication is up again. If the user agrees, the user exit invokes the java class asking it to post a [near real time message](#).
 - The user exit calculates a unique outgoing message ID for the external system. The message id is calculated using a database sequence whose name is referenced in the external option **Message ID Database Sequence Name** on the external system table. This message id is passed to the real time message engine to be populated as an NDS context entry to support an [asynchronous response to the message](#).
 - The user exit passes the NDS type that references the Appointment Cancel - Near Real Time condition flag. The FA Id is also passed to the real time message engine to be posted as an NDS context entry. A response to this message will be an acknowledgement. No further updates to field activity data are expected.

If no error is received, the appointment page continues with the "change" action. The appointment period is unlinked from the field activity and if no other field activities are linked to this appointment period, the appointment period is deleted. In addition, the field activity characteristics for the reservation number and forced appointments are removed from the FA.

NOTE:

The above updates are performed even if the message is sent near real time and no response has been received. This was done to cancel the appointment in our system to cater for the situations when the customer wants to rebook the appointment prior to receiving the acknowledgement from the external system.

Validating Meter / Item Installations

Incoming Validate Meter / Item Message

When Oracle Utilities Customer Care and Billing receives the incoming message to validate a meter or item, the service, called CIPOVMIP, which processes the message, creates a pending notification download staging record (using the NDS type whose notification download condition is Validate Meter/Item). It creates context records for the Badge Number being validated along with an indication of whether the badge number is for a Meter or an Item.

FASTPATH:

For more information about NDS types available in product integrations, refer to Oracle Utilities Mobile Workforce Management NDS Types and Oracle Utilities Work and Asset Management NDS Types listed in the documentation on integration.

The system provides an XAI Inbound Service, `ValidateMeterItemRequest`, defined to invoke this service. The following table lists the fields to populate for this service.

General Information
Message ID
FA ID
FA External ID
Meter / Item Flag (M or I)
Badge Number

Outgoing Validate Meter / Item Message

The service provided with the product that processes the Validate Meter / Item notification download staging record does the work of validating the badge number. The service, called `CIPOVRSP`, does the following:

- Finds a unique device (meter or item) corresponding to the badge number provided.

NOTE:

The validation is only possible if the badge number is unique for a meter or item. If multiple values are found, a negative acknowledgement is returned.

- Verifies that the device is not retired
- Verifies that the Meter type or Item type of the device being verified is defined as valid for the SP Type associated with the field activity's service point.
- It verifies that the device is not already installed somewhere else.
- If the device is a meter, it verifies that there is an effective Meter Configuration for the meter on or before the message date/time.

NOTE:

Return Meter Configuration Type. The meter configuration type and the collection of the meter's registers are included in the output record returned to Oracle Utilities Mobile Workforce Management/Oracle Utilities Work and Asset Management.

- If the device is an item, it verifies that the item's receive date is on or before the message date.

The system provides an XAI Inbound Service, `ValidateMeterItemResponse`, defined to invoke this service. The following table lists the fields populated by this service.

General Information
Message ID
FA ID
FA External ID

Meter / Item Flag (M or I)

Badge Number

StatusFlag (Y - positive acknowledgement, F - negative acknowledgement)

Meter Config Type (populated only for meters)

Collection of registers (populated only for meters)

ErrorCode (populated only for negative acknowledgement)

Error Message (populated only for negative acknowledgement)

Setting Up The System To Enable FA Integration

The following section provides an overview of how to enable FA integration with an external system.

Service Provider Setup

In order to use XAI to interface with an external system, you must define a service provider. Once the service provider is defined, you must design your outbound messages.

Defining Characteristic Types For FA Integration

The following characteristic types must be defined to facilitate FA integration.

External System Task Characteristic Type

If you have decided to map the tasks on your external system to the Oracle Utilities Customer Care and Billing field activity types, then you need to define the characteristic types based on your decisions. It is possible to map a combination of fields from the external system to a field activity type in Oracle Utilities Customer Care and Billing.

- Refer to the following topics in the documentation on integration for more information about how you should define characteristics in product integrations:
 - Oracle Utilities Mobile Workforce Management Characteristic Types
 - Oracle Utilities Work and Asset Management Characteristic Types
- Include Field Activity Type in the characteristic entity collection

Forced Appointments Characteristic Type

If your implementation supports [forced appointments](#), the appointment booking logic attempts to store a characteristic on a field activity with a forced appointment.

- Create an ad hoc [characteristic type](#).
- Include Field Activity in the characteristic entity collection

Reservation Characteristic Type

If your external system defines a separate reservation ID for appointments, the appointment booking logic attempts to store a characteristic on a field activity with the appointment reservation number.

- Create an ad hoc [characteristic type](#).
- Include Field Activity in the characteristic entity collection

Override Phone Characteristic Type

Create a characteristic type for override phone if your implementation supports capturing a contact phone number on the field activity.

- Create an ad hoc [characteristic type](#).
- Include Field Activity in the characteristic entity collection

Field Activity ID Characteristic Type

The sample FA integration algorithm may be configured to populate the field activity ID as a characteristic on any NDS records it creates. This facilitate in drilling down from the NDS record to the field activity. To support this logic,

- Create a foreign key [characteristic type](#) (if you don't already have one defined for Field Activity Id).
- Include Notification Download Staging in the characteristic entity collection

Setting Up Outbound Messages

The sample integration provided with the base product includes a predefined list of messages that are sent to an external system under various conditions. The messages are generated either from the sample FA integration algorithm or the sample real time appointment interaction. In each case, an NDS type is required to define properties of the message. Rather than hard-coding an NDS type, the integration algorithm and the user exit that manages the real time appointment interaction use a Notification Download Condition to reference the NDS type.

At implementation time, you should define an appropriate NDS type for each notification download condition listed below if it is applicable to your business.

The following download conditions are used in the [sample FA integration algorithm](#):

- FA Cancellation
- FA Creation
- FA Changed
- FA Rescheduled
- Appointment Cancel via FA Cancel

For each of the above NDS types, you must reference the following context types: Field Activity ID, Message ID. They should also reference an XAI inbound service that has been defined for the [Extract FA Info](#) service. The system provides an XAI inbound service called ExtractFAInfo, which you may use.

NOTE:

You can refer to the demonstration data provided with the system to view samples for pre-configured NDS types and their condition flags.

The following download conditions are used in the sample [real time appointment interaction](#):

- Get Available Appointments
- Appointment Book
- Appointment Cancel

The above NDS types do not need to reference a real XAI inbound service because the real time appointment interface is responsible for building the XML request.

NOTE:

You can refer to the demonstration data provided with the system to view samples for pre-configured NDS types and their associated XAI inbound services.

The following download conditions are used to send [real time appointment messages in near real time](#).

- Appointment Book - Near Real Time
- Appointment Cancel - Near Real Time

Because the XML request is built by the appointment interface prior to the creation of the NDS, these NDS types should reference a special XAI inbound service called CDxProcessXDS. This service basically tells the download staging sender that the XDS already exists and doesn't need to be created.

Designing Your External System Feature Configuration

For each external system, you must define a [feature configuration](#) with a feature type of FA Integration.

Note that it is also possible for you to define multiple entries in the feature configuration table for a single external system. You would do this if your external system may be configured in multiples ways for different dispatch groups. For example, maybe your service territory includes urban areas and rural areas. Perhaps your rules for scheduling appointments differ based on the location of the premise. You could define two separate feature configurations and define the appropriate appointment options for each one. When defining your dispatch groups, be sure to define separate dispatch groups based on the operations area and link the appropriate feature configuration accordingly.

If you define multiple feature configurations, consider whether they should all reference the same service provider. One consideration is whether or not the method of communication with the external system is the same for all feature configurations.

Configure the options for your external system interaction.

NOTE:

Your implementation may define additional options types. You do this by add new lookup values to the lookup field WFM_OPT_TYP_FLG.

Option	Description
Account Rel. Type - Company Contact	Identify the account relationship type used to define a company's contact person.
Allow Forced Appointments	Use this option to indicate if forced appointments are supported. Possible values are Y and N.
Allow Manual Appointment	Use this option to indicate if a user is allowed to manually set up an appointment . Possible values are Y and N.
Allow Manual Appointment Cancellation	Use this option to indicate if a user is allowed to manually cancel an appointment . Possible values are Y and N.
Allow Multiple Reservations	Use this option to indicate if booking appointments for multiple field activities is allowed. Possible values are Y and N.
Allow Narrowing Of Appointment Window	Use this option to indicate if the user is allowed to further narrow down a selected appointment window . Possible values are Y and N.
Appointment Forced Characteristic Type	When an appointment reservation is forced, a characteristic of this type is added to the field activity. Note that the field activity's FA type must also define this as a valid characteristic type.

Appointment Java Class Interface	This is the java class implementation used to interface with the external system to support real time appointment interaction .
Default Days of Available Appointment	This option is used to determine the end date of the search period when choosing a dispatch group on the appointment page.
Hi-Low Review	Use this option to indicate if meter reads coming from the external system should be reviewed for Hi-Low failures and trended. Possible values are Y and N.
Intermediate Status to Prevent FA Cancel	This option is used to identify Intermediate Status values that should prevent the system from automatically canceling a Field Activity . The value entered here should correspond to a valid lookup value for the field FA_INT_STATUS_FLG.
Intermediate Status to Skip Message	This option is used to identify FA Intermediate Status value used when a Field Activity is created by an external system or when other information for a field activity is updated by an external system. The base FA integration algorithm uses this information to ensure that messages sent to the external system to highlight new field activities or changes to field activities are only triggered when additions / changes are initiated in our system. The value entered here should correspond to a valid lookup value for the field FA_INT_STATUS_FLG.
Message ID Database Sequence Name	<p>The name of the database sequence to be used to get the next unique message ID for this external system. This is used to facilitate an asynchronous response to the message.</p> <p>If you interface with more than one external system, you may choose to use the same sequence name for all external systems or to define a separate sequence name for each external system. If you choose to define multiple feature configuration records for the same service provider, be sure that each feature configuration references the same sequence name because generated message IDs must be unique for the service provider.</p> <p>The base product provides the database sequence CI_WFM_MSGID_SEQ, which may be referenced here.</p>
No Address Premise Type	Indicate the premise type used to identify a premise that is used for field activities that are not related to a specific premise .
Phone Number Characteristic Type	Indicate the characteristic type used to identify an override phone number on the field activity.
Phone Type - Business	Indicate the phone type used to identify a business phone number.
Phone Type - Fax	Indicate the phone type used to identify a fax number.
Phone Type - Home	Indicate the phone type used to identify a home phone number.
Plant Source	Some external systems require a reference to a Plant in our system. There are several options for where an implementation may define this value. This option is used to identify where the Plant is defined. - enter FECO if the plant field is defined in Feature Configuration - enter OPAR if the plant field is defined in the SP Operations Area - enter SPCH if the plant field is defined in the SP Characteristic
Plant Value	If the Plant Source is FECO enter the value of the Plant. If the Plant Source is OPAR enter the field service class used to identify the plant value on the SP operations area. If the Plant Source is SPCH enter the characteristic type used to identify the plant value on SP characteristic. Only one option value may be defined for a given feature configuration.

Reservation Characteristic Type	When an appointment is successfully booked for the field activity, the external system often assigns a unique reservation number to the appointment. This reservation number is linked to the field activity as a characteristic using this characteristic type . Note that the field activity's FA type must also define this as a valid characteristic type.
Service Provider	This is the service provider defined for your external system.
To Do Type for FA Response	Indicate the To Do type to use to create a To Do entry . The system supplies the To Do type TD-FARSP that may be plugged-in here. The program populates the To Do Entry with the sort keys, drill keys and message parameters as shown in this base package To Do. If you want to create your own To Do Type, you must set up the values to match those in the base To Do Type.
User Defined Criteria Field	This is used on the appointment page to add specific appointment selection criteria . The value of this option should reference a Field defined in the system metadata. The appointment page allows up to 10 user defined criteria fields.
User Defined Result Field	This is used on the appointment page to add specific appointment selection result information . The value of this option should reference a Field defined in the system metadata. The appointment page allows up to 10 user defined result fields.

For each message that may be received from an external system, map the external system message to an internal system message. Refer to [Feature Configuration - Messages](#) for more information.

Designing Your External System Field Activity Types

For each type of field activity that is interfaced to an external system, create an appropriate FA type.

- Indicate that the FA type is eligible for dispatch.
- Configure the appropriate value for Appointment Booking based on your business requirements.
 - If you have decided to map your external system tasks to the Oracle Utilities Customer Care and Billing field activity type, then for each field activity type, create one or more characteristics to identify how it is mapped to the equivalent task in the external system.
- Indicate the FA characteristics that are valid for field activities of this type
 - If your external system allows [forced appointments](#), define the forced appointment characteristic type created above and referenced on your external system feature configuration.
 - If your external system defines a separate reservation ID for appointments and this FA type allows appointments, define the reservation characteristic type created above and referenced on your external system feature configuration.

Refer to the following topics in the documentation on integration for more information about how you should use characteristics in product integrations:

- Oracle Utilities Mobile Workforce Management Characteristic Types
- Oracle Utilities Work and Asset Management Characteristic Types

Designing Your Dispatch Groups

When a field activity is created, the system uses the [Field Service Control](#) to assign the field activity to a dispatch group based on the type of activity, the type of service point and the operations area that manages the service point. If the dispatching for this service point is managed by an external system, the [dispatch group](#) should be configured to interface with the external system:

- The dispatch group references the appropriate external system feature configuration.
- You must indicate an appropriate [FA integration](#) algorithm.

Considerations When Switching To External System Integration

If your implementation is currently using the field order functionality and is planning to switch to interface field activities to an external system, here are some considerations.

Field orders are not required when integrating field activities to an external system. As a result you may choose to disable the field order related functionality:

- The automatic dispatch background process (FOD) and download field order background processes (FDS) no longer need to be scheduled. In addition, the printing processes FODL and DSGPFODL no longer need to be scheduled.
- The menu items [Field Order](#), [Group Premise FAs](#) and [Field Order Search](#) are no longer applicable. Consider disabling security for these pages.

When switching over to begin using an external system, you will undoubtedly have pending field activities that need to be interfaced to the external system. If you change the field activity's dispatch group from one that does not reference an external system to one that does, the sample [field activity integration algorithm](#) will generate an FA Creation message to the external system.

It is possible that your pending field activities are already linked to field orders. If that is the case, you will not be able to change the dispatch group on the field activity. The recommendation for switching to an external system for dispatching is to change your pending field activities to remove the link between the field activity and the field order. As mentioned above, the field order is no longer needed. Once you remove the link then you are able to change the dispatch group on the field activity.

If you prefer to leave the field order / field activity link in place then you must change the dispatch group on your field order to one that references the new external system.

Outage System Integration

The following section describes functionality provided for the integration between Oracle Utilities Customer Care and Billing and Oracle Utilities Network Management System.

The Big Picture of Outage System Integration

Oracle Utilities Customer Care and Billing is the central repository for customer information; for example, name, address, phone number, etc. Oracle Utilities Network Management System is the central repository for outage information; for example, outage calls, affected supply nodes, expected restoration time, etc.

In an integrated environment, each system provides information to the other system so that they can operate together seamlessly.

- The outage system uses the set of current customers to determine and manage outages to minimize their impact
- The outage system is informed of outages captured in Oracle Utilities Customer Care and Billing
- Oracle Utilities Customer Care and Billing uses the current status of an outage at a given premise for customer service

Customer Information Integration

The outage system needs information about current customers to determine and manage outages to minimize their impact. The current customer information in Oracle Utilities Customer Care and Billing must be made available in Oracle Utilities Network Management System. This can be done via data synchronization.

FASTPATH:

Refer to [The Big Picture of Sync Requests](#) for more information about synchronizing data.

Interfacing Outage Calls

The following points describe the integration:

- Oracle Utilities Customer Care and Billing is able to record trouble calls for a particular service point that exists in the system, as well as for an unknown service point, i.e. a fuzzy call. For a fuzzy call, the caller must provide either a street intersection, or a street segment.
- When an outage call is created and sent to the external system, an algorithm on the outage call business object is responsible for creating an outbound message that's sent to the external system. This is a real-time synchronous interface between Oracle Utilities Customer Care and Billing and Oracle Utilities Network Management System.
- Oracle Utilities Network Management System processes the Calls table and creates Incidents.

FASTPATH:

Refer to the *Oracle Utilities Customer Care and Billing - Network Management System Integration Implementation Guide* for information about outage call integration.

Outage Inquiry

Oracle Utilities Customer Care and Billing provides query transactions that can be used to make real-time synchronous calls to NMS and inquire on one of the following:

- Job History for a particular customer, service point, location or call identifier
- Call History for a particular customer, service point, location or call identifier
- Planned Outage Jobs for a particular service point

FASTPATH:

Refer to the *Oracle Utilities Customer Care and Billing - Network Management System Integration Implementation Guide* for information about outage query integration.

Setting Up The System To Enable Outage Integration

The following section provides an overview of how to enable the integration between Oracle Utilities Customer Care and Billing and Oracle Utilities Network Management System.

External System Setup

An external system must be setup in order integrate Oracle Utilities Network Management System. Once the external system is defined, specify this on the one NMS Integration feature configuration so the system knows which external system to use for outage queries.

Define Outbound Message Types

The following outbound message types are required for the integration:

- An outbound message type is required for each of the outage queries available
 - Job History
 - Call History
 - Planned Outages
- Once the outbound message types are defined, specify this on the one NMS Integration feature configuration so the system knows which outbound message types to use for outage queries.
- In addition, an outage call outbound message type is required for sending outage calls to Oracle Utilities Network Management System. This outbound message type must be referenced on your outage call types.

Define Characteristic Types

The following characteristic types must be defined to facilitate Outage integration:

Outage Group Code Characteristic Type

These characteristics are used to describe the outage problem.

- Create at least one pre-defined characteristic type
- For each characteristic type, define its list of valid values
- Include Service Task Type in the characteristic entity collection

NOTE:

Characteristic Type Prefix. The system attempts to build a dropdown list of your valid outage group codes when maintaining outage group types. To achieve this, all outage group code characteristic types must use the same prefix. This prefix must be defined on the NMS Integration feature configuration.

Integration with Outage Management. The outage codes must be defined in both Oracle Utilities Customer Care and Billing and the outage management system. Refer to your Oracle Utilities Network Management System documentation for information about defining the outage codes there.

Contact Name Characteristic Type

This is used to link the contact name of the caller associated with an outage call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call Contact Name Characteristic Type feature option

Contact Number Characteristic Type

This is used to link the contact number of the caller associated with an outage call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call Contact Number Characteristic Type feature option

Call Identifier Characteristic Type

This is used to link the call identifier supplied by the caller associated with an outage call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call Identifier Characteristic Type feature option

Street Name Characteristic Type

This is used to link location information supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call Street Name Characteristic Type feature option

Cross Street Name Characteristic Type

This is used to link location information supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call Cross Street Name Characteristic Type feature option

Block Number Characteristic Type

This is used to link location information supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call Block Number Characteristic Type feature option

City Characteristic Type

This is used to link location information supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call City Characteristic Type feature option

State Characteristic Type

This is used to link location information supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call.

- Create an ad-hoc [characteristic type](#)
- Include Service Task Type in the characteristic entity collection
- Specify this characteristic type on the one NMS Integration feature configuration using the Outage Call State Characteristic Type feature option

NMS Integration - Feature Configuration

Create a [feature configuration](#) with the type NMS Integration. Populate entries for all the options.

NOTE:

Only one. The system expects only one NMS Integration feature configuration to be defined.

Configure the options for your interaction with the outage system.

NOTE:

Your implementation may define additional options types. You do this by adding new lookup values to the lookup field NMS_OPT_TYP_FLG.

Option	Description
External System	This defines the external system used on outbound messages created when querying outage information in NMS from the outage management information portal page. Refer to External System Setup for more information. Only one value is allowed for this option.
Outbound Message Type - Call History	This defines the outbound message type used on outbound messages created when querying outage call history in NMS from the outage management information portal page. Refer to Define Outbound Message Types for more information. Only one value is allowed for this option.
Outbound Message Type - Job History	This defines the outbound message type used on outbound messages created when querying outage job history in NMS from the outage management information portal page. Refer to Define Outbound Message Types for more information. Only one value is allowed for this option.
Outbound Message Type - Planned Outages	This defines the outbound message type used on outbound messages created when querying planned outages in NMS from the outage management information portal page. Refer to Define Outbound Message Types for more information. Only one value is allowed for this option.
Outage Group Code Characteristic Type Prefix	The system uses this prefix to populate the outage group code dropdown list during trouble call processing . Refer to Define Characteristic Types for more information.

Only one value is allowed for this option.

Schema Constants - Feature Configuration

Create a [feature configuration](#) with the type Schema Constants. Populate entries for all the options listed below.

NOTE:

Only one. The system expects only one Schema Constants feature configuration to be defined.

Configure the options for your interaction with the outage system.

NOTE:

Your implementation may define additional options types. You do this by adding new lookup values to the lookup field F1CN_OPT_TYP_FLG.

Option	Description
Outage Call Contact Name Characteristic Type	This is used to link the contact name of the caller associated with an outage call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.
Outage Call Contact Number Characteristic Type	This is used to link the contact number of the caller associated with an outage call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.
Outage Call Identifier Characteristic Type	This is used to link the call identifier supplied by the caller associated with an outage call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.
Outage Call Street Name Characteristic Type	This is used to link the street name supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.
Outage Call Cross Street Name Characteristic Type	This is used to link the cross street name supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.
Outage Call Block Number Characteristic Type	This is used to link the block number supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.
Outage Call City Characteristic Type	This is used to link the city supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call. Refer to Define Characteristic Types for more information. Only one value is allowed for this option.

Outage Call State Characteristic Type

This is used to link the state supplied by the caller associated with a fuzzy trouble call as a characteristic on the outage call. Refer to [Define Characteristic Types](#) for more information.

Only one value is allowed for this option.

Business Flags

It is possible that information detected in one product may be useful or even critical to share with another product. The framework provides functionality for receiving information from an external system that acts as a type of flag or alert that may need investigation. This allows any system to store detected business flags in a common way and share that information with one or more other systems. See [Business Flags](#) for more information.

This section provides further information about this.

Service Point Business Flags

Overview

This product has implemented business flags that represent situations that exist at a customer's service point that are shared across external applications; such as, meter data management, mobile work force management, and analytics applications. The situations that a business flag can represent ranges from critical knowledge that requires manual analysis for resolution to purely informational notifications.

This sharing of information allows users of each system to quickly understand the status of a service point as it relates to situations that users of the system would benefit from. For example, the analytics system may identify potential theft situations that should be investigated prior to releasing the next bill.

The subsequent sections describe business flag functionality that is specific to the service point business flag as implemented by this product.

See [About Business Flags](#) in the *Oracle Utilities Framework Administration User Guide* for a detailed understanding of the components of business flags.

Business Flags Can Require Analysis

For more serious situations a business flag can require manual analysis from a user. These types of business flags will generate a To Do when they are received from the external application. To resolve the To Do a user will investigate the service point based on the type of business flag present. Generally this investigation will result in the business flag being either confirmed or rejected. To reach the conclusion of whether the business flag should be confirmed or denied the user can be provided with a set of available actions that can be performed on the business flag.

See [CI-BFANL](#) for more information.

Business Flag Available Actions

Allowed actions are implemented as BPA scripts that will guide the user through a particular method of investigating or resolving the business flag. The following actions are supported:

- **Create Field Activity** - This will initiate a field activity associated to the business flag's service point. The type of field activity will be selected by the user.
- **Create Case** - This will initiate a case associated to the business flag's service point. The type of case will be selected by the user.

- **Create Notification** - This will initiate communication with the customer for the business flag's service point. The type of communication will be determined by the notification type selected by the user.

Business Flag Adding Available Actions

To extend the actions that are available to the user for investigating a business flag the following steps can be taken:

1. A new BPA script should be developed that provides the functionality for your particular business use case
2. That BPA script will be added to the transactional business object for the business flag using the option type "Valid Action"
3. Edit the business flag type to include the newly created action in the "Allowed Actions" section of the business flag type

Making Allowed Actions Available

Allowed actions are only accessible on those business object statuses that have been configured to support them. To permit access to the available actions on a given status one must add a business object status option for the type "Execution of Valid Actions Permitted" with a value of "Y".

Business Flag Can Prevent Bills from Freezing

Some business flags, particularly those that require investigation, should place a hold on the billing process to prevent potentially incorrect bills from being issued to the customer. When certain types of business flags are present on a service point it indicates that there may be a problem with the reported usage that is underlying the bill and as such it may not be desirable to allow the billing process to complete until the investigation into that usage has been completed. For example, if the business flag indicated potential theft or abnormally high usage it may be preferable to delay issuing the bill until an investigation can be performed and the issue rejected or resolved.

This functionality can be enabled through configuration of the Prevent Bill Segment Freeze Due to Business Flag algorithm. See algorithm type Prevent Bill Segment Freeze Due to Business Flag ([C1-PRVBSFBB](#)) for more detail.

A Field Activity Can Determine Business Flag Confidence

When a business flag results in a field activity being issued and a field team is sent to investigate the service point the result of that field activity investigation will drive the resolution of the business flag. This is done through the field remarks left by the field team. The field activity remarks can either be defined as confirming the business flag or rejecting it. By selecting the appropriate field activity remark the field team will conclude the investigation into the business flag and the final confidence will be updated as appropriate.

For those field activity remarks that should either confirm or reject the business flag the Business Flag Confidence Update algorithm type should be provided for the field activity remark – activation plug-in spot.

Please refer to algorithm type Business Flag Confidence Update ([C1-BSFLGCFUP](#)) for more detail.

Business Flags Can Create Alerts

Some business flags have a high enough importance that they should be highlighted to any user that is viewing data for the service point the business flag is related to. Whether a business flag should create an alert is controlled on the business flag type. Furthermore, the appropriate algorithm must be configured in the "Installation Options – Framework" "Algorithms" tab for the "Control Central Alert" system event.

The product delivers the algorithm type "Highlight Open Business Flags"

Please refer to algorithm type *Business Flag Confidence Update* ([C1-BSFLGCFUP](#)) for more detail.

Customer Relationship Management Integration

Oracle Utilities Customer Care and Billing provides tools to facilitate the integration with customer relationship management applications such as Oracle Social Relationship Management (SRM).

The following sections describe this functionality.

About Customer Relationship Requests

A customer relationship request is used to process requests that are initiated from an external customer relationship management system.

The following sections highlight customer relationship request functionality.

Social Relationship Management Requests

Oracle Social Relationship Management (SRM) is an example of an external system that can initiate customer relationship requests.

Social network posts cover a vast range of topics, a number of which could be related to the users' concerns with their utility service. For example:

- A Facebook user could post on a community page about a power outage in a specific area.
- A Twitter user might post about specific problems with his/her utility service — e.g. high bill complaints.

Social conversations of this nature can be monitored and managed via Oracle SRM. Oracle SRM's Engage module allows its users to see all social conversations, identify customer-specific issues and route the messages to the appropriate systems/channels.

Requests from social relationship management systems typically include:

- Information about the original posting on the social network/community — e.g. author's identifier, link to the post, etc.
- Information about the posting within the social relationship management system — e.g. the user that sent the request to the utility, link to the post, etc.
- Additional details that will help with processing/resolution — e.g. customer/account numbers, address, bill IDs, etc.

Creating Customer Relationship Requests

Most customer relationship requests will be initiated from an external system.

The base product provides an Add Customer Relationship Request (C1-AddCustRR) inbound web service that can be used in a customer relationship management integration to create customer relationship request records. This inbound web service invokes a processing script that is configured in Customer Relationship Integration master configuration. The base product supplies an example of this processing script. It maps the inbound service's input data into the customer relationship request record. Refer to the C1-CreCustRR service script system data for processing details.

Setting Up Customer Relationship Request Configuration

The following topics highlight the general configuration steps required to use customer relationship request functionality and to integrate with a customer relationship management application such as Oracle Social Relationship Management (SRM).

Customer Relationship Management Integration Configuration

The integration with a customer relationship management application such as Oracle Social Relationship Management (SRM) requires the setup of master configuration data that controls the processing of your customer relationship requests.

To set up customer relationship integration master configuration, navigate using **Admin > General > Master Configuration**. You are brought to an all-in-one portal with options to add/edit the Customer Relationship Integration master configuration record. Refer to the inline help for more details on how each section in this record is configured.

Defining Customer Relationship Request Types

A customer relationship request type defines properties that control how a customer relationship request is processed.

Refer to [About Customer Relationship Requests](#) for an overview of customer relationship request functionality.

To maintain customer relationship request types, open **Admin > Integration > Customer Relationship Request Type > Search**.

This is a standard [All-in-One portal](#).

The information captured on the customer relationship request type depends on the business objects supported by your implementation. Refer to the inline help text for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI_CUST_REL_REQ_TYPE](#).

Maintaining Customer Relationship Requests

This section describes the functionality supported for viewing and maintaining customer relationship requests.

Refer to [About Customer Relationship Requests](#) for an overview of customer relationship request functionality.

Navigate using **Main > Integration > Customer Relationship Request > Search**. You are brought to a query portal with options for searching for customer relationship requests.

Once a customer relationship request record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Customer Relationship Request** zone provides basic information about a customer relationship request. Refer to the inline help for more information.

The **Notes** zone allows users to capture additional notes to the customer relationship request.

Control Table Setup Sequence

The topics in this section describe the order in which the control tables should be set up.

Core Control Table Setup Sequence

To implement the system, you must set up your organization's business rules in "control tables". Setting up these tables is time-consuming because we allow you to tailor many aspects of the system to meet your organization's requirements. We strongly recommend that you take the time to document how you plan to set up all of these tables before you use the following roadmap to enter the core control data. Time spent understanding the interrelationships between this data will reap the rewards of a clean system that meets your current and long term needs.

While we describe the transactions and options in more detail in other sections of this manual, use the following chart (and the remaining sections of this chapter) as your roadmap. Here we list the order in which you perform tasks and the pages you'll use to set up your system. The order is important because some information must exist before other information can be defined (i.e., many dependencies exist).

NOTE: Auto setup. The Auto Setup column in the following table contains suggestions to save you time. It also indicates if a control table contains information when the system is installed.

NOTE: You don't have to set up every control table. You need only set up those control tables that govern functions that are applicable to your organization.

Function	Menu	Auto Setup
Global Context		
Algorithm	Admin, Algorithm. You will need to set up an algorithm that populates global context values. The global context is used by various zones in the system to display relevant data. This algorithm is plugged-in on the installation record .	
Accounting Environment		
Country & State	Admin, Country	
Currency Codes	Admin, Currency Code	USD is automatically populated
Accounting Calendar	Admin, Accounting Calendar	
GL Division	Admin, General Ledger Division	
Security Environment		
Application Service	Admin, Application Service	All base package transactions are automatically populated
Security Type	Admin, Security Type	
User Group	Admin, User Group Note, you won't be able to set up users at this point	One user group, ALL_SERVICES, is automatically setup. It references all other application services and a single user called SYSUSER.
Language	Admin, Language	ENG is automatically populated
Display Profile	Admin, Display Profile	Two display profiles are automatically setup: NORTHAM displays currencies and dates in a classic American format; EURO displays information in a classic European format
Data Access Role	Admin, Data Access Role	
Access Group	Admin, Access Group	
User	Admin, User	SYSUSER is automatically set up. Note that you may import your users (and user groups) from an external source .
Return to User Group	You must return to your user groups and define all of their users	
Customer Class Environment		

Customer Class	Admin, Customer Class. At this point, you'll only be able to set up your customer class codes. You will return to these customer classes throughout the setup process to populate additional information.
<hr/>	
Financial Transaction Environment	
Work Calendar	Admin, Work Calendar
CIS Division	Admin, CIS Division
Revenue Class	Admin Revenue Class
Algorithm	Admin, Algorithm. You will need to set up the algorithm that constructs a distribution code's corresponding GL account when it is interfaced to the general ledger
Distribution Code	Admin, Distribution Code
Bank & Bank Accounts	Admin, Bank
Billable Charge Template	Admin, Billable Charge Template. Note, if you want the system to default service quantities onto billable charges created using this template, you must setup the appropriate unit of measure code, time-of-use code and/or service quantity identifier.
Billable Charge Upload Line Type	Admin, Billable Charge Line Type
Algorithm	Admin, Algorithm. You will need to set up several algorithms. These algorithms: 1) retrieve a bill segment's consumption, 2) calculate a bill segment's bill lines, 3) construct a bill segment's financial transaction, 4) cancel previously estimated bill segments
Bill Segment Type	Admin, Bill Segment Type
Algorithm	Admin, Algorithm. You may want to set up an algorithm that formats the Bill Segment information that is displayed throughout the system for a specific Bill Segment Type. This algorithm is plugged-in on the Bill Segment Type.
Algorithm	Admin, Algorithm. You will need to set up the algorithm that constructs a payment segment's financial transaction
Payment Segment Type	Admin, Payment Segment Type
Algorithm	Admin, Algorithm. You will need to set up the algorithm that constructs an adjustment's financial transaction
Algorithm	Admin, Algorithm. Several plug-in spots are available to perform additional logic when processing adjustments. For example, if you have the system calculate adjustments, you must set up an adjustment generation

algorithm. Refer to [Adjustment Type](#) for other available plug-in spots that may be used by your implementation.

Algorithm	Admin, Algorithm. You may want to set up an algorithm that formats the Adjustment information that is displayed throughout the system for a specific Adjustment Type. This algorithm is plugged-in on the Adjustment Type .
Algorithm	Admin, Algorithm. You may want to set up an algorithm that formats the Adjustment information that is displayed throughout the system. This algorithm is plugged-in on the installation record .
Adjustment Type	Admin, Adjustment Type
Adjustment Type Profile	Admin, Adjustment Type Profile
Approval Profile	Admin, Approval Profile. Note, an approval profile references a To Do type and one or more To Do Roles; these must be set up before you can set up the approval profile. After the approval profile(s) are set up, they must be referenced on the adjustment types that they govern.
Cancel Reason - Bill	Admin, Bill Cancel Reason
Cancel Reason - Payment	Admin, Payment Cancel Reason
Cancel Reason - Adjustment	Admin, Adjustment Cancel Reason
Tender Type	Admin, Tender Type
Tender Source	Admin, Tender Source
A/P Request Type	Admin, A/P Request Type
Issuing Center	Admin, Issuing Center. You will need to set up issuing centers if your organization assigns document numbers to bills.
Installation	Admin, Installation Options - Framework and Admin, Installation Options. Many fields on the installation record impact the financial transaction environment. Refer to the description of the Billing and Financial Transaction tabs and the Messages tab in the Framework page for more information.
Algorithm	Admin, Algorithm. You will need to set up an algorithm that distributes payments.
Algorithm	Admin, Algorithm. You will need to set up an algorithm that handles overpayment situations.
Algorithm	Admin, Algorithm. You may need to set up an algorithm if specific customers can have individual bill due dates.

Algorithm	Admin, Algorithm. You may need to set up an algorithm if you want the system to delete bills that contain only information about historical payments.
Algorithm	Admin, Algorithm. You may need to set up an algorithm if you want the system to levy a non-sufficient funds charge if a payment is canceled due to non-sufficient funds.
Algorithm	Admin, Algorithm. You will need to set up an algorithm that formats the bill information that is displayed throughout the system. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. You will need to set up an algorithm that formats the payment information that is displayed throughout the system. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. You will need to set up an algorithm that defaults the amount when a payment is manually added. This algorithm also calculates the amount of an automatic payment for a bill for an account with an active auto pay option. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. Refer to Customer Class for other available plug-in spots that may be used by your implementation to perform additional logic when processing payments and bills.
Return to Customer Class	Admin, Customer Class. You will need to plug-in the algorithms defined above on your customer classes.
Budget Environment	
Algorithm	Admin, Algorithm. You will need to set up several algorithms at this time: How To Calculated The Recommended Budget Amount, How To Periodically True Up A Customer's Budget Amount, The Circumstances When The System Should Highlight A Customer As Having An Anomalous Budget.
Budget Plan	Admin, Budget Plan
Algorithm	Admin, Algorithm. Budget eligibility is set at the SA type level. You will need to set up an override budget eligibility algorithm if some service agreements for an SA type are not eligible for budget based on certain conditions.
Customer Environment	

Account Management Group	Admin, Account Management Group. Note, you will probably have to set up To Do Type and To Do Roles before you can setup account management groups. Refer to Assigning A To Do Role for more information on how account management groups may be used to define an entry's role.
Account Relationship	Admin, Account Relationship Type
Alert Type	Admin, Alert Type
Bill Message	Admin, Bill Message
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a bill in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your bill image software.
Bill Route Type	Admin, Bill Route Type
Contract Quantity Type	Admin, Contract Quantity Type
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a letter in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your letter image software.
Letter Template	Admin, Letter Template
Customer Contact Class	Admin, Customer Contact Class
Customer Contact Type	Admin, Customer Contact Type
Conservation Programs	Admin, Conservation Program. You will need to set up conservation programs if your organization provides rebates to customers based on eligibility and verification of newly purchased appliances and hardware that are rated to conserve the demand for energy.
Algorithm	Admin, Algorithm. You may need to set up the algorithms that determine if person ID's are in a predefined format.
Identifier Type	Admin, Identifier Type
SICs	Admin, SIC Code
Tax Exempt Type	Admin, Tax Exempt Type
Algorithm	Admin, Algorithm. You may need to set up the algorithms that determine if phone numbers are in a predefined format.
Phone Type	Admin, Phone Type.
Person Relationship Type	Admin, Person Relationship Type.
Algorithm	Admin, Algorithm. You will need to set up an algorithm that formats the person information that is displayed throughout the system. This

	algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. You will need to set up an algorithm to validate a person's name. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. You can override the system's standard account information string by setting up an algorithm that produces this string of information. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a letter in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a bill in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation record .
Installation	Admin, Installation Options. Many fields on the installation record impact the Customer Environment. Refer to the description of the Main , Person , and Account tabs for more information.
Statements	
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a statement in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your statement image software.
Statement Route Type	Admin, Statement Route Type
Statement Cycle	Admin, Statement Cycle
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a statement in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation record .
Automatic Payment (EFT) Environment	
Algorithm	Admin, Algorithm. You will need to set up an algorithm to create automatic payments. This algorithm is plugged-in on the installation record .
Tender Source	Admin, Tender Source

Note: earlier, you created tender sources for the remittance processor and your cash drawers. At this point, you'll need to add at least one tender source for automatic payments. Why? Because automatic payments get linked to a tender control (which, in turn, gets linked to a tender source) when they are interfaced out of the system.

Algorithm	Admin, Algorithm. You will need to set up the appropriate automatic payment date calculation algorithm to populate the extract, GL interface and payment dates on automatic payments.
Auto Pay Route Type	Admin, Auto Pay Route Type
Tender Type	Admin, Tender Type Note: earlier, you created tender types for things like cash, checks, etc. At this point, you'll need to add a tender type for each type of automatic payments (e.g., direct debt, credit card, etc.).
Work Calendar	Admin, Work Calendar. You need only set up additional work calendars if the auto pay sources (i.e., the financial institutions) have different working days than does your organization
Algorithm	Admin, Algorithm. If you need to validate the customer's bank account or credit card number, you will need to set up the appropriate validation algorithms.
Auto Pay Source Type	Admin, Auto Pay Source Type
Algorithm	Admin, Algorithm. You may need to set up an algorithm if your customers can define a maximum withdrawal limit on their autopay options.
Return to Customer Class	Admin, Customer Class. You should plug-in the Autopay Over Limit Algorithm in each appropriate customer class.
Deposit Environment	
Algorithm	Admin, Algorithm. You will need to set up several algorithms at this time: The Definition Of A Good Customer, When To Refund A Deposit To A Customer, When To Recommend An Additional Deposit, How / When To Calculate Interest, How To Generate The Recommended Deposit Amount.
Deposit Class	Admin, Deposit Class
Non Cash Deposit Type	Admin, Non Cash Deposit Type
Field Work Environment - Phase 1	

Representative	Admin, Representative
Operation Area	Admin, Operation Area
Field Service Class	Admin, Field Service Class
Algorithm	Admin, Algorithm. You will need to set up the algorithms that execute special functions (if any) when a field activity is completed
Field Activity Type & Steps	Admin, Field Activity Type
Field Activity & Field Order Cancellation Reason	Admin, Fieldwork Cancel Reason
Field Activity & Field Order Reschedule Reason	Admin, Fieldwork Reschedule Reason
Algorithm	Admin, Algorithm. If you need anything special to happen when a remark is associated with a field activity (e.g., generate a To Do), you will need to set up an algorithm to do whatever you need to do and associate it with the respective Field Activity Remark.
Field Activity Remarks	Admin, Field Activity Remark
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a field order in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your field order image software.
Dispatch Group	Admin, Dispatch Group
Disconnect Location	Admin, Disconnect Location
Field Activity Type Profiles	Admin, Field Activity Type Profile
Algorithm	Field activities for Start / Stop will only be created if the SA Type linked to the service point has a SASP field work creation algorithm. Refer to the SA Type section below.
Algorithm	Admin, Algorithm. If you have software that's capable of reconstructing an image of a field order in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation record .
Credit & Collections Environment (if you collect on overdue bills (as opposed to overdue debt), you will NOT set up these tables; refer to Overdue Processing - Set Up Tasks for the list of control tables required to collect on overdue bills)	
Algorithm	Admin, Algorithm. You may need to set up algorithms if you have non-standard collection events.
Collection Event Type	Admin, Collection Event Type
Algorithm	Admin, Algorithm. You may need to set up a collection process cancellation algorithm if your organization allows individual

service agreements to be removed from a collection process if they are paid (rather than performing cancellation based on all SAs in a debt class).

Collection Process Template	Admin, Collection Process Template
Collection Class	Admin, Collection Class
Algorithm	Admin, Algorithm. You will need to set up several algorithms at this time: Collection process cancellation criteria, Severance process cancellation criteria, and Override arrears due to pay plans.
Debt Class	Admin, Debt Class
Write Off Debt Class	Admin, Write Off Debt Class
Algorithm	Admin, Algorithm. You will need to set up Collection Condition algorithms.
Collection Class Control	Admin, Collection Class Control
Algorithm	Admin, Algorithm. You may need to set up algorithms if you have non-standard severance events.
Severance Event Type	Admin, Severance Event Type
Algorithm	Admin, Algorithm. You may need to set up a severance process cancellation algorithm if your organization allows a severance process to be canceled when the related service agreement is paid (rather than performing cancellation based on all SAs in a debt class).
Severance Process Template	Admin, Severance Process Template
Algorithm	Admin, Algorithm. You will need to set up several algorithms at this time: How to refer debt to a collection agency, How to transfer debt to another active service agreement, How to write down small amounts of debt, and How to refund credit balances to a customer.
Algorithm	Admin, Algorithm. You may need to set up algorithms if you have non-standard write-off events.
Write Off Event Type	Admin, Write Off Event Type (Note, you'll have to wait until you have defined your SA Types before you can set up the Write Off Events because SA Type is a necessary parameter to write off debt).
Write Off Process Template	Admin, Write Off Process Template
Write Off Control	Admin, Write Off Control
Collection Agency	Admin, Collection Agency. Note, each collection agency references a person therefore you must set up a person for each agency before you can enter collection agency information.

Algorithm	Admin, Algorithm. You may need to set up algorithms if you have special logic that should be executed when a pay plan is canceled.	
Pay Plan Type	Admin, Pay Plan Type	
Payment Method	Admin, Payment Method	
Third Party Payor	Admin, Third Party Payor. Note, you must create an account before you can create a third party payor.	
Installation	Admin, Installation. Several fields on the installation record impact the Credit & Collections Environment.	
Algorithm	Admin, Algorithm. You will need to setup an algorithm that's called when a user write-off debt real time.	
Return to Customer Class	Admin, Customer Class. You should plug-in the Autopay Over Limit Algorithm in each appropriate customer class.	
Services & Characteristics		
Service Type	Admin, Service Type	
Algorithm	Admin, Algorithm. If you have ad hoc characteristic types, you may need to set up the algorithms that control how they are validated	
Foreign Key Reference	Admin, FK Reference. If you have foreign key characteristic types, you may need to set up foreign key references to control how the user selects the characteristic values (and how the foreign key values are validated).	All base package FK references are automatically populated
Characteristic Type & Values	Admin, Characteristic Type	
Device Testing Environment		
Algorithm	Admin, Algorithm. If you need to validate a specific device test component result, you will need to set up the appropriate validation algorithms.	
Device Test Component Type	Admin, Device Test Component Type	
Algorithm	Admin, Algorithm. If you need the system to determine if a device test's results are considered "passing", you will need to set up an algorithm to perform this processing.	
Device Test Type	Admin, Device Test Type	
Algorithm	Admin, Algorithm. If you have the system select meters / items for testing, you will need to set up an algorithm to perform this processing.	
Meter & Item Environment		

Meter Type	Admin, Meter Type [Note - you won't be able to define the collection of valid Equipment Types and Item Types until after you define the Item Types. You also will not be able to define the collection of Meter Configuration Types until after you define the Meter Configuration Types.]
Meter ID Type	Admin, Meter ID Type
Manufacturer / Model	Admin, Manufacturer
Unit of Measure	Admin, Unit of Measure
Time of Use	Admin, Time of Use
Meter Configuration Type	Admin, Meter Configuration Type
Retirement Reason	Admin, Retire Reason
Protocol	Admin, Protocol
Read Out Type	Admin, Read Out Type
Algorithm	Admin, Algorithm. You will need to set up an algorithm to generate estimated consumption.
Trend Area	Admin, Trend Area
Trend Class	Admin, Trend Class
High / Low	Admin, High Low Factor
Algorithm	Admin, Algorithm. You will need to set up an algorithm that calculates the high / low limits used on meter reads. This algorithm is plugged-in on the installation record .
Meter Location	Admin, Meter Location
Meter Read Instructions	Admin, Meter Read Instruction
Algorithm	Admin, Algorithm. If you need anything special to happen when a meter read with a given remark is uploaded (e.g., generate a field activity), you will need to set up an algorithm to do whatever you need to do and associate it with the respective Meter Read Remark.
Meter Reader Remarks	Admin, Meter Reader Remark
Meter Read Source	Admin, Meter Read Source
Meter Read Warning	Admin, Meter Read Warning
Item Type	Admin, Item Type
Algorithm	Admin, Algorithm. You will need to set up an algorithm to format the standard meter info that appears throughout the system. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. You will need to set up an algorithm to format the standard item info that appears throughout the system. This

algorithm is plugged-in on the [installation record](#).

Premise & Service Point Environment		
Premise Type	Admin, Premise Type	
Algorithm	Admin, Algorithm. You will need to set up an algorithm to format the standard premise info that appears throughout the system. This algorithm is plugged-in on the installation record .	
Algorithm	Admin, Algorithm. You will need to set up an algorithm that formats the service point information that is displayed throughout the system. This algorithm is plugged-in on the installation record .	
Algorithm	Admin, Algorithm. You may need to set up the algorithms that determine if geographic ID's are in a predefined format.	
Geographic Type	Admin, Geographic Type	
Service Point Type	Admin, SP Type. [Note - you won't be able to define the SP Type's SA Types until after you define the SA Types or the FA Type Profiles until after you define the Field Activity Type Profiles.]	
Facility Level 1 to 2	Admin, Facility Level 1 to 2	
Facility Level 2 to 3	Admin, Facility Level 2 to 3	
Field Work Environment - Phase 2		
Field Service Control	Admin, Field Service Control	
Bill & Service Cycle Environment		
Bill Cycle, Bill Cycle Schedule	Admin, Bill Cycle	
Bill Period, Bill Period Schedule	Admin, Bill Period	
Route Type	Admin, Route Type	
Service Cycle / Route	Admin, Service Cycle	
Service Schedule	Admin, Service Schedule	
Rate Environment		
Frequency	Admin, Frequency	
Service Quantity Identifier	Admin, Service Quantity Identifier	
Algorithm Type	Admin, Algorithm Type. If you create new pre-processing calculation rules, you must set up an algorithm type for each such rule (the algorithm type defines the types of parameters that are passed to the calculation rule).	All base package algorithm types are automatically populated
Bill Factor	Main, Rates, Bill Factor	
Algorithm Type	Admin, Algorithm Type. If you create new Register Rules you must set up an algorithm type for each such rule (the algorithm type	All base package algorithm types are automatically populated

defines the types of parameters that are passed to the register rule).

Rate	Main, Rates, Rate Schedule
Calculation Group	Main, Rates, Calculation Group
Algorithm	Admin, Algorithm. If you use algorithms to dynamically change step boundaries, calculate prices, or implement calculation rule eligibility rules, you must set up these algorithms.
Calculation Rule	Main, Rates, Calculation Rule
Bill Factor Value	Main, Rates, Bill Factor Values
Bill Factor Interval Values	Main, Rates, BF Interval Values
Item Type SQ Estimate	Admin, Item Type SQ Estimate
Degree Day	Admin, Degree Days
Late Payment Environment	
Algorithm	Admin, Algorithm. You will need to set up the algorithm that determine if customers in a customer class are eligible for late payment charges
Algorithm	Admin, Algorithm. You will need to set up the algorithm that levies late payment charges for customers in a customer class
Return to Customer Class	Admin, Customer Class. You will need to plug-in the late payment charge algorithms set up above.
SA Configuration	
Algorithm	Admin, Algorithm. You will need to set up the algorithms that determine: <ul style="list-style-type: none">• How to calculate the late payment charge amount for service agreements of a given type• Special criteria to be tested before a service agreement is severed.• How to create field activities for service agreements of a given type.• Special processing that should take place prior to the completion of a bill that references service agreements of a given type.• Special processing that should take place during completion of a bill that references service agreements of a given type.• Special processing that should take place when service agreements of a given type are created.• Special processing that should take place when a financial transaction is frozen for service agreements of a given type.

Algorithm	Admin, Algorithm. You may want to set up an algorithm that formats the SA information that is displayed throughout the system. This algorithm is plugged-in on the installation record .
Algorithm	Admin, Algorithm. You may want to set up an algorithm that formats the SA information that is displayed throughout the system for a specific SA Type. This algorithm is plugged-in on the SA Type .
Algorithm	Admin, Algorithm. If you want a Control Central alert to highlight when the current account has any stopped service agreement(s), you will need to set up the algorithm that does this. This algorithm is plugged-in on the installation record .
Service Agreement Type	Admin, SA Type
Terms and Conditions	Admin, Terms and Conditions
SA Type Start Options	Admin, SA Type Start Option
Update SP Types with initial SA types and with FA Type Profiles	Admin, SP Type
SA Relationships	
SA Relationship Type	Admin, SA Relationship Type
Service Provider	Admin, Service Provider. Note, you must create a person before you can create a service provider. If you have financial relationships (you bill for them or they bill for you), you must also create an account and a financial settlement service agreement before you can create the service provider.
SA Type / SA Relationship	Admin, SA Type SA Relationship Type
Notification and Workflow	
Workflow Event Type	Admin, Workflow Event Type
Workflow Process Template	Admin, Workflow Process Template
Notification Upload Type	Admin, Notification Upload Type
Workflow Process Profile	Admin, Workflow Process Profile
Notification External (Sender) ID's	Admin, Notification External ID
Notification Download Type	Admin, Notification Download Type
Service Provider.	Admin, Service Provider. Note, you must create a person before you can create a service provider.
Notification Download Profile	Admin, Notification Download Profile
Algorithm	Admin, Algorithm. If you want a Control Central alert to highlight when the current account and/or premise has active workflow processes, you will need to set up the

algorithm that does this. This algorithm is plugged-in on the [installation record](#).

Sales and Marketing	
Order Hold Reason	Admin, Order Hold Reason
Order Cancel Reason	Admin, Order Cancel Reason
And more...	Refer to Campaign and Package Setup Sequence for additional setup requirements
Service Credit Membership	
Algorithm	Admin, Algorithm. You may need to set up algorithms for the service credit membership type and service credit event type to control behavior for the service credit membership and its events.
Credit Unit	Admin, Credit Unit. If your service credits record non-monetary units.
Service Credit Membership Type	Admin, Service Credit Membership Type
Service Credit Event Type	Admin, Service Credit Event Type
Membership Inactive Reasons	Admin, SC Membership Inactive Reason
Wrap Up	
Algorithm	Admin, Algorithm. You will need to set up the algorithms that determine: #x2022; Special alerts on Control Central (assuming you have special alerts)
Installation Options	Admin, Installation Options - Framework and Admin, Installation Options. At this point, it's a good idea to double-check everything on the installation record.
Postal Default	Admin, Postal Code Default

If you have cash drawers you will also need to set up the following information:

- Create a person / account to which you will link your over / under service agreement. Refer [How To Get An Unbalanced Tender Control In Balance \(Fixing Over/Under\)](#) for more information.
- Create a service agreement to which your over/under payments will be linked. This service agreement will reference your over / under SA type. Refer to [Over / Under Cash Drawer Segmentation](#) for more information.

If you upload payments from an external source (e.g., a remittance processor or lock box), you must set up the following information:

- Create a person and account to which the system will link payments with invalid account. Refer to [Phase 3 - Create Payment Events, Tenders, Payments and Payment Segments](#) for information about the process that books invalid payments to this account. Refer to [How To Transfer A Payment From One Account To Another](#) for how a user transfers the payment from the invalid account to the correct account (once known).
- Create a service agreement for this account. This service agreement will reference your payment suspense SA type. The system needs this service agreement so that it can distribute the invalid account's payment (and this is necessary so that cash will reflect the payment). Refer to [Payment Upload Error Segmentation](#) for more information.
- Update the tender source associated with the respective source of payments to indicate the service agreement created in the previous step should be used for payments with invalid accounts. Refer to [Setting Up Tender Sources](#) for more information.

- Because the payment upload process simply books payments that reference invalid accounts to the account associated with the suspense service agreement on the payment's tender source, this account should belong to a customer class with the appropriate payment distribution algorithms. This may entail creating a new customer class that will only be used on suspense accounts. This customer class would need the following algorithms:
- We'd recommend using a simple payment distribution algorithm like [PYDIST-PPRTY](#) (distribute payment based on SA type's payment priority).
- We'd recommend using an overpayment distribution algorithm like [OVRPY-PPRTY](#) (distribute overpayment to highest priority SA type).

The remaining sections describe additional control tables that must be set up for specific functional areas.

Cross-Reference To The Remaining Chapters

The table in the previous section describes the order in which you should enter your control tables. These tables are described at length in the following chapters.

- Refer to [Defining General Options Addendum](#) and [Defining General Framework Options](#) for a discussion of the control tables associated with general functionality (e.g., country codes, state codes, etc.).
- Refer to [Defining Financial Transaction Options](#) for a discussion of the tables affecting your financial transactions (e.g., bill segment types, payment segment types, etc.)
- Refer to [Defining Customer Options](#) for a discussion of the control tables affecting persons, accounts and service agreements.
- Refer to [Defining Fieldwork Options](#) for a discussion of the control tables affecting fieldwork.
- Refer to [Defining Credit and Collections Options](#) for a discussion of the control tables affecting your collection activities.
- Refer to [Defining Meter and Item Options](#) for a discussion of the control tables affecting your meters and items.
- Refer to [Defining Premise and Service Point Options](#) for a discussion of the control tables affecting your premises and service points.
- Refer to [Defining Cycles and Schedules](#) for a discussion of the control tables affecting your cyclical processes.
- Refer to [Rates](#) for a discussion of the control tables affecting your rates.
- Refer to [Defining SA Type Options](#) for a discussion of the control tables affecting your service agreement types.
- Refer to [Defining Background Process](#) for a discussion of the control tables affecting your background processes.
- Refer to [Defining Algorithms](#) for a discussion of the control tables affecting the algorithms referenced on many control tables.
- Refer to [Defining SA Relationships](#) for a discussion of the control tables affecting the relationships between service providers.
- Refer to [Defining Workflow and Notification Options](#) for a discussion of the control tables affecting the processing of notifications to and from service providers.
- Refer to [Defining Interval Billing Options](#) for a discussion of the control tables affecting the interval billing options for your customers.
- Refer to [Statements](#) for a discussion of the tables affecting the statement setup options for your customers.
- Refer to [Defining Service Credit Options](#) for a discussion of the tables affecting the service credit membership setup options for your customers.

Open-Item Accounting Table Setup Sequence

Open-item accounting tables need only be set up if your organization practices [Open Item Accounting](#). Refer to [Setting Up The System To Enable Open Item Accounting](#) for a description of the tables that must be set up to enable this functionality.

Fund Accounting Table Setup Sequence

Fund accounting tables need only be set up if your organization practices [Fund Accounting](#). Refer to [Setting Up The System To Enable Fund Accounting](#) for a description of the tables that must be set up to enable this functionality.

Payment Event Distribution Table Setup Sequence

Payment event distribution tables need only be set up if your organization opted to use the distribution rules method to create payment events. Refer to [Setting Up The System To Use Distribution Rules](#) for a description of the tables that must be set up to enable this functionality.

Loans Table Setup Sequence

Loans need only be set up if your organization offers [loans](#) to your customers. Refer to [Setting Up The System To Enable Loans](#) for a description of the tables that must be set up to enable this functionality.

Quotes Table Setup Sequence

Quotes need only be set up if your organization sends quotes to customers or prospects. Refer to [Defining Quotation Options](#) for a description of the tables that must be set up to enable this functionality.

Non-billed Budget Table Setup Sequence

[Non-billed budgets](#) need only be set up if your organization allows your customers to pay set amounts at specified intervals (e.g. every two weeks). Refer to [Setting Up The System To Enable Non-billed Budgets](#) for a description of the tables that must be set up to enable this functionality.

Appointments Table Setup Sequence

[Appointments](#) need only be set up if your organization allows your customers to make appointments for field activities. Refer to [Enabling Appointments](#) for a description of the tables that must be set up to enable this functionality.

Scripting Table Setup Sequence

Scripts need only be set up if your organization opts to create [scripts](#) to step your users through business processes. Refer to [Defining Script Options](#) for information about scripting and the tables that must be set up to enable this functionality.

Reports Setup Sequence

In order to use the reporting tool, you will need to set up reporting options. Refer to [Configuring The System To Enable Reports](#) for more information.

XML Application Integration Setup Sequence

In order to use the XAI tool for sending information between third parties, you will need to set up XAI control tables. Refer to [XML Application Integration](#) for more information.

Case Management Setup Sequence

Case management options need only be set up if your organization uses cases to manage issues. Refer to [Setting Up Case Types](#) for more information.

Workforce Management Setup Sequence

Workforce management options need only be set up if your organization interfaces with an external workforce management system. Refer to [Setting Up The System To Enable FA Integration](#) for more information.

Umbrella Agreement Management Setup Sequence

Umbrella agreement management options need only be set up if your organization uses umbrella agreements to manage contracts. Refer to the integration documentation for more information.

Outage Management Setup Sequence

Outage management options need only be set up if your organization interfaces with Oracle Utilities Network Management System. Refer to the integration documentation for more information.

Prepaid Metering Setup Sequence

Prepaid metering options need only be set up if your organization offers prepaid metering service to your customers. Refer to [Defining Prepaid Metering Options](#) for more information.

Batch Scheduler Setup Sequence

Batch scheduler options need only be set up if your organization uses the batch scheduling functionality provided by the system rather than a third party batch scheduling system. Refer to [Setting Up The Batch Scheduler](#) for more information.

Zone Set Up

Most zones are delivered with the base-package and do not require any configuration. However, some zones are only available if configured by your implementation. Refer to [Configuring Zones](#) for more information.

To Do Options Setup

Refer to [Setting Up To Do Options](#) for more information on how to configure the system to match your organization's To Do management needs.

The Conversion Tool

This section describes the Oracle Utilities Customer Care and Billing conversion tool.

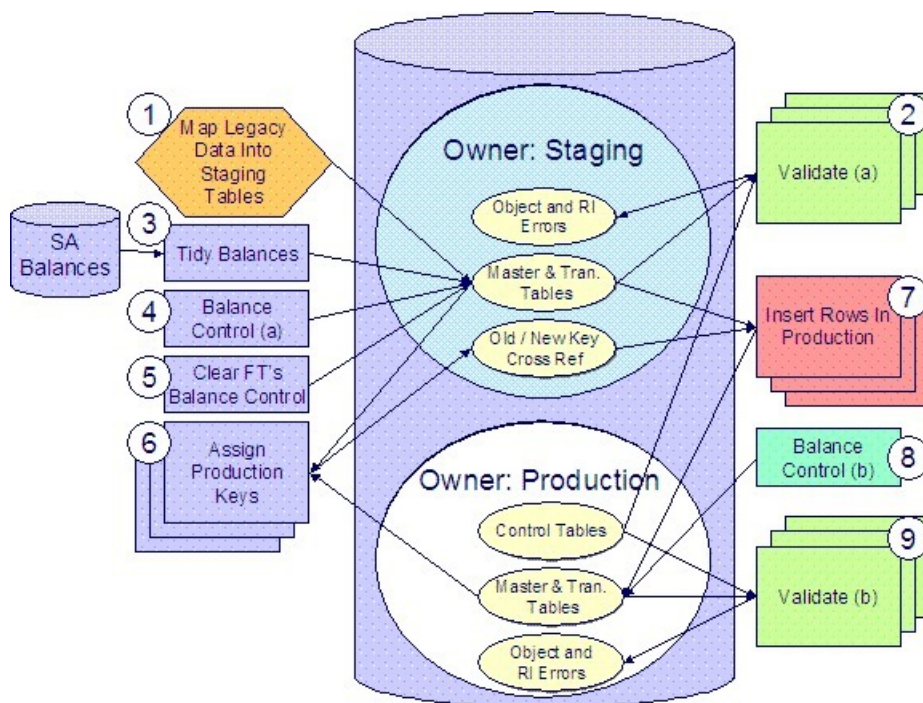
Introduction

When you're ready to convert data from your legacy system into Oracle Utilities Customer Care and Billing, you will have analyzed your CIS processing requirements according to your business and organizational needs and set up the control tables accordingly.

FASTPATH:

Refer to the **Administration Guide** for a complete discussion of the various control tables and the order in which they must be set up.

After the control tables are set up, you are ready to load data into the system from your legacy system. This conversion effort involves several steps as illustrated in the following diagram:



The following points briefly outline each of the above tasks:

- **Map Legacy Data Into Staging.** During this step, your legacy master data (e.g., account, person, premise, meter) and transaction data (e.g., bills, payments, meter reads) is migrated into the system. Notice that you are not migrating this data directly into production. Rather, your rows are loaded into tables that are identical to the production tables; they just have a different owner. Refer to [Appendix B - Multiple Owners In A Single Database](#) for information about table ownership.

WARNING:

The above diagram illustrates how the system is configured to support the conversion effort in the standard installation, i.e., the staging tables are in the same database as the production tables (each with a different owner). However, it is possible for the staging tables to be in a separate database. This option requires additional effort on your part (because you would have to copy the control tables from production into your staging database). Please refer to [Appendix B - Multiple Owners In A Single Database](#) for information about this alternative.

Mapping legacy data into the system is probably the most challenging part of the conversion process because the system is a normalized database (and most legacy applications are not).

- **Validate (a).** During the validation (a) step, the system validates the data you loaded into the staging tables. Two types of validation programs exist:
 - **Object Validation Programs.** Each of the system's master data objects (e.g., person, account, premise, meter, etc.) is validated using the same logic that is used to validate data added by users in your production system.
 - **Referential Integrity Validation Programs.** After you have successfully validated the master data objects, the referential integrity validation programs are executed to validate transaction data and to highlight "orphaned" rows. These programs check the validity of the foreign keys on all rows on all tables.

NOTE:

Control tables from production. It's important to notice that the validation programs validate your staging data using the control tables that have been set up in production. Refer to [Appendix B - Multiple Owners In A Single Database](#) for a description of how this works.

- **Tidy Balances.** During this step, the system creates adjustments that cause each SA's current and payoff balances to equal the desired balances. The desired balances are supplied on a flat file prepared by you.
- **Balance Control (a).** During this step, you run the balance control program and then verify that the balances that it generates are consistent with the balances in your legacy system.
- **Clear FT's Balance Control.** In the previous step, the system creates a balance control and links it to the FT's. If the balance control's balances are consistent with the amount of receivables being transferred into the system, you should run the Clear FT's Balance Control program. This program simply resets the Balance Control column on the FT so that the FT's can be included in a balance control (see the last step below) after they have been transferred to production.
- **Assign Production Keys.** During this step, the system allocates random, clustered keys to the rows in the staging database.
- **Insert Rows Into Production.** During this step, the system populates your production tables with rows from the staging. When the rows are inserted, their prime keys are reassigned using the data populated in the previous step.
- **Balance Control (b).** During this step, you run the balance control program against production. You do this to verify the balances in production are consistent with the values of receivables converted from your legacy application.
- **Validate (b).** During this step, you rerun the object validation programs, but this time against production. We recommend rerunning these programs to confirm that the insertion programs have executed successfully. We recommend running these programs in random sample mode (e.g., validate every 1000th object) rather than conducting a full validation in order to save time. However, if you have time, you should run these programs in full validation mode (to validate every object).

Conversion Tool Steps

The following sections provide more details about the steps in the conversion process.

Map Legacy Data Into Staging Tables

This section provides some high level discussion about mapping legacy data to the system's staging tables. Refer to [The Staging Tables](#) for details about the staging tables in the system.

NOTE:

Recommendation. You can use any method you prefer to load Oracle Utilities Customer Care and Billing data from your legacy application. However, we recommend that you investigate your database's mass load utility (as opposed to using insert statements) as the mechanism to load the staging tables. In addition, we strongly recommend that you disable the indexes on these tables before populating these tables and then enable the indexes after populating these tables.

Populating Characteristic Tables. There are many maintenance objects that include a characteristic table used to capture miscellaneous information about the object, e.g. Person, Account, Service Agreement, etc. Most of these tables include an indexed column used when searching by characteristic value called Search Characteristic Value. During conversion and depending on the type of characteristic, this column must be populated as follows:

- **Predefined:** Populate search characteristic value with the contents of the characteristic value column converted to upper case.
- **Ad hoc:** Populate search characteristic value with the first 50 bytes of the ad hoc characteristic value column converted to upper case.
- **Foreign key:** Populate search characteristic value by concatenating the values of each foreign key characteristic value column to a maximum of 50 bytes.

A Note About Keys

The prime keys of the tables in the staging database are either system-assigned random numbers or they aren't. Those tables that don't have system-assigned random numbers have keys that are a concatenation of the parent's prime-key plus one or more additional fields.

Every table whose prime key is a system-assigned random number has a related table that manages its keys; we refer to these secondary tables as "key tables". The following points provide more information about the key tables:

- Key tables are used by programs that allocate new keys. For example, before a new account ID is allocated, the key assignment program checks the account key table to see if it exists.
- Key tables only have two columns:
 - The key of the object.
 - An environment ID. The environment ID identifies the database in which the object resides.
- Key tables are named the same as their primary table with a suffix of "_K". For example:
 - The key table for CI_ACCT is CI_ACCT_K
 - The key table for CI_PREM is CI_PREM_K
- The name of every table's key table is defined under the Generated Keys column in the Table Names sections in [The Staging Tables](#).
- When you populate rows in tables with system-assigned keys, you must also populate a row in the related key table. For example, if you insert a row into CI_ACCT, you must also insert a row into CI_ACCT_K. The environment ID of these rows must be the same as the environment ID on this database's [installation record](#).
 - When you populate rows in tables that reference this record as a foreign key, you must use the appropriate key to ensure the proper data relationships. For example, if you insert a row in CI_SA for the above account, the ACCT_ID column must contain the temporary account key.
- When you insert rows into your staging database, the keys do not have to be random, system-assigned numbers. They just have to be unique. A later process, [Allocate Production Keys](#), will allocate random, system-assigned keys prior to production being populated.

Validate Information In The Staging Tables

During the first validation step, the system validates the data you loaded into the staging tables. Two types of validation programs exist:

- **Object Validation Programs.** The object validation programs validate each of the system's master data objects (e.g., person, account, premise, meter, etc.) and a limited number of transaction data objects (e.g., field activity, field order, etc.). Please note that these programs call the same programs that are used to validate data added by users in your production system.
- **Referential Integrity Validation Programs.** After the master data objects have been validated, the referential integrity validation programs are executed to validate transaction data and to highlight "orphaned" rows. These programs simply check the validity of the foreign keys on all rows on all tables.

The contents of this section describe how to execute the validation programs.

Object Validation Programs

Each of the objects described under [Master Data](#) must be validated using the respective object validation program indicated in its Table Names section.

In a limited number of cases object validation is available for [Transaction Data](#) objects, where customers may convert transaction data that is still pending. For example, if you are converting pending field activities, you want to ensure that the data is valid. For these cases you may also be converting historic records. For example, in addition to the pending field activities you are converting completed field activities to keep a historic view. You may not want to perform validation on completed records. As a result the background processes provided for transaction data allow you to limit the validation to records in a give status.

We strongly recommend validating each object in the following steps:

- Execute each object's validation program in random-sample mode to highlight pervasive errors. When you execute a validation in random-sample mode, you are actually telling it to validate every X records (where X is a parameter that you supply to the job). Refer to [Submitting Object Validation Programs](#) for more information about the parameters supplied to these background processes.
- You can view errors highlighted by validation programs using the [Validation Error Summary](#) transaction.
- Correct the errors using SQL. Note, you can use the base package's transactions (e.g., Person Maintenance, Premise Maintenance, etc.) to correct an error if the error isn't so egregious that it prevents the object from being displayed on the browser.
- After all pervasive errors have been corrected; re-execute each object's validation program in all-instances mode to highlight elusive, one-off errors. Refer to [Submitting Object Validation Programs](#) for more information about the parameters supplied to these background processes.

WARNING:

Whenever an object validation program is run, it is necessary to delete all previously recorded errors associated with its tables from the validation error table before it inserts new errors.

After the various object validation programs run cleanly, run the referential integrity validation programs as described in the next section.

NOTE:

Another use for these programs. In addition to validating your objects after conversion or an upgrade, the validation programs are another use. Say for example, you want to experiment with changing the validation of a person and you want to determine the impact of this new validation on your existing persons. You could change the validation and then run the person validation object - it will produce errors for each person that fails the new validation.

Submitting Object Validation Programs

The object validation programs that are described in the [staging tables](#) table names matrices are classic background processes as they can also be run against production data. You submit these processes in the same way you submit any background process in production.

Referential Integrity Validation Programs

It's important to understand that only master data objects (e.g., persons, accounts, meters, premises, etc.) are validated by the object validation programs discussed above. This means that only master data objects will have their foreign keys checked for validity by the object validation programs. You must run the referential integrity programs to validate all other data.

The referential integrity validation programs highlight:

- Orphaned rows because orphan rows, by definition, don't reference an object.
- Invalid foreign keys on transaction data.

NOTE:

Validating Transaction Data. You may wonder why transaction data is not subject to the object validation routines. This is because: a) the production system only needs validation logic for master data because transaction data is not entered by users, and b) most conversions necessitate loading skeletal transaction data because the legacy system typically doesn't contain enough information to create accurate transactions in the system.

Each of the tables described under [Transaction Data](#) must be validated using the respective referential integrity validation program indicated in its Table Names section. We strongly recommend validating each table in the following steps:

- Execute each table's referential integrity validation program. Refer to [Submitting Referential Integrity Validation Programs](#) for more information about the parameters supplied to these background processes.
- You can view errors highlighted by this process using the [FK Validation Summary](#) transaction.
- Correct the errors using SQL (you cannot use the application to correct these types of errors).
- Rerun the referential integrity programs until no errors are produced.

WARNING:

Whenever you run a referential integrity validation program, it deletes all errors associated with its table from the referential integrity error table.

In order to highlight orphaned rows in the master data, run the referential integrity validation programs against all tables described under [Master Data](#) using the procedure described above.

When ALL referential integrity programs indicate the staging database is clean, you may now proceed to the next step - [tidy balances](#).

Submitting Referential Integrity Validation Programs

The referential integrity validation programs described under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices) are submitted using a batch driver program, CIPVRNVB, and this program is executed in the staging database. Please note that the referential integrity validation programs may also be run in the production environment on occasion, to determine the integrity of data in the production database.

You should supply the following parameters to this program:

- **Batch code.** The batch code associated with the appropriate table's referential integrity validation program. Refer to each table listed under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices) for each referential integrity batch code / program.
- **Batch thread number.** Thread number is not used and should be left blank.
- **Batch thread count.** Thread count is not used and should be left blank.
- **Batch rerun number.** Rerun number is not used and should be left blank.
- **Batch business date.** Business date is the date supplied to the referential integrity validation programs and the date under which statistics will be logged.
- **Total number of commits.** Total number of commits is not used and should be left blank.
- **Maximum minutes between cursor re-initiation.** Maximum minutes between cursor re-initiation is not used and should be left blank.
- **User ID.** User ID is only used to log statistics for the execution of the batch job.

- **Password.** Password is not used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start (Y/N), trace program exit (Y/N), trace SQL (Y/N) and output trace (Y/N).** These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed.

Recommendations To Speed Up Validation Programs

The following points describe ways to accelerate the execution of the validation programs:

- Ensure that statistics are recalculated after data has been inserted into the staging tables. For Oracle users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.
- [Object validation programs](#) should be run multi threaded.
- Execute shorter running validation processes (e.g., less records) first so that the error data can be analyzed while other processes are busy running.
- [Referential integrity validation programs](#) run fairly quickly without much tuning. However, additional benefits are gained by running several programs at the same time.
- Remember that the [object validation programs](#) can be run in "validate every n th row". We recommend running these programs using a largish value for this parameter until the pervasive problems have been rectified.

Tidy Balances

This background process creates adjustments that cause each SA's current and payoff balances to equal its balance in the legacy system (note: the batch control ID of CNV-BAL is used for this process).

NOTE:

Submitting this process. You submit this process in the staging database. Refer to [Tidy Balances](#) for a description of this process and its parameters.

You supply the desired balances to this background process in a flat file in the following format:

Field	Size	Description
SA ID	X10	The unique identifier of the service agreement
Payoff Balance Sign	X1	Positive or Negative value indicates debit or credit balance respectively
Payoff Balance	N15.2	The SA's payoff balance (how much the customer really owes).
Current Balance New Charge Sign	X1	Positive or Negative value indicates debit or credit balance respectively
Current Balance - New Charge	N15.2	The amount of the SA's current balance that is considered a new charge, i.e., it hasn't started aging yet.
Current Balance Amount 1 Sign	X1	Positive or Negative value indicates debit or credit balance respectively

Current Balance - Amount 1	N15.2	The amount of the SA's current balance that is X days old (X is defined in the next field)
Age of Current Balance - Amount 1	N3	The number of days old the prior field is (if you keep your debt in "buckets" as opposed to knowing the exact number of days it has aged, you will have to choose an exact age). Set this value to zero if the value of amount 1 should be considered a "new charge" (i.e., it should only start aging when it is swept onto the next bill produced for the SA's account)
Current Balance Amount 2 Sign	X1	Positive or Negative value indicates debit or credit balance respectively
Current Balance - Amount 2	N15.2	The amount of the SA's current balance that is X days old (X is defined in the next field)
Age of Current Balance - Amount 2	N3	The number of days old the prior field is (if you keep your debt in "buckets" as opposed to knowing the exact number of days it has aged, you will have to choose an exact age)
Current Balance Amount 3 Sign	X1	Positive or Negative value indicates debit or credit balance respectively
Current Balance - Amount 3	N15.2	The amount of the SA's current balance that is X days old (X is defined in the next field)
Age of Current Balance - Amount 3	N3	The number of days old the prior field is (if you keep your debt in "buckets" as opposed to knowing the exact number of days it has aged, you will have to choose an exact age)
Current Balance Amount 4 Sign	X1	Positive or Negative value indicates debit or credit balance respectively
Current Balance - Amount 4	N15.2	The amount of the SA's current balance that is X days old (X is defined in the next field)
Age of Current Balance - Amount 4	N3	The number of days old the prior field is (if you keep your debt in "buckets" as opposed to knowing the exact number of days it has aged, you will have to choose an exact age)
Current Balance Amount 5 Sign	X1	Positive or Negative value indicates debit or credit balance respectively
Current Balance - Amount 5	N15.2	The amount of the SA's current balance that is X days old (X is defined in the next field)
Age of Current Balance - Amount 5	N3	The number of days old the prior field is (if you keep your debt in "buckets" as opposed to knowing the exact number of days it has aged, you will have to choose an exact age)

NOTE:

Submitting this process. You submit this process in the staging database. Refer to [Tidy Balances](#) for a description of this process and its parameters.

Balance Control (a)

During this step, you run the balance control programs and then verify that the balances that it generates are consistent with the balances in your legacy system.

NOTE:

Submitting this process. You submit this process in the staging database. Refer to [The Big Picture of Balance Control](#) for more information about the balance control processes. Refer to [Balance Control](#) for information about the page used to view the balances generated by this process.

Clear FT Balance Control

In the previous step, the system created a balance control and links it to the FT's. If the balance control's balances are consistent with the amount of receivables being transferred into the system, you should run the Clear FT's Balance Control program. This program simply resets the Balance Control column on the FT so that the FT's can be included in a balance control (see the last step below) after they have been transferred to production. Note: the batch control ID of CNV-BCG is used to request this process.

NOTE:

Submitting this process. You submit this process in the staging database. Refer to [Reset Balances](#) for a description of this process and its parameters.

Allocate Production Keys

The topics in this section describe the background processes used to assign production keys to the staging data.

The Big Picture of Key Assignment

It's important to understand that the system does not overwrite the prime-keys on the rows in the staging database, as this is a very expensive IO transaction. Rather, a series of tables exist that hold each row's old key and the new key that will be assigned to it when the row is [transferred into the production database](#). We refer to these tables as the "old key / new key" tables. The old key / new key tables are named the same as their primary table, but rather than being prefixed by "CI", they are prefixed by "CK". For example, the old key / new key table for CI_ACCT is called CK_ACCT.

The insertion programs that transfer the rows into the production database use the new key for the main record of the key along with any other record where this key is a foreign key. Note that the capability of assigning the new key to a foreign key applies to

- "True" foreign keys, i.e. where the key is a column in another table. For example, CI_SA has a column for ACCT_ID.
- FK reference characteristics. These are characteristics that define, through an FK reference, the table and the key that this characteristic represents.

The insertion programs are not able to assign "new keys" to foreign keys defined in an XML structure field (CLOB).

The key assignment programs listed under [Master Data](#) and [Transaction Data](#) (in the table names sections) are responsible for populating the old key / new key tables (i.e., you don't have to populate these tables). Because the population of the

rows in these tables is IO intensive, we have supplied detailed instructions that will accelerate the execution time of these programs.

NOTE:

Why are keys reassigned? The conversion process allocates new prime keys to take advantage of the system's parallel processing and data-clustering techniques in the production system (these techniques are dependent on randomly assigned, clustered keys).

Iterative conversions. Rather than perform a "big bang" conversion (one where all customers are populated at once), some implementations have the opportunity to go live on subsets of their customer base. If this describes your implementation, please be aware that the system takes into account the existing prime keys in the production database before it allocates a new key value. This means when you convert the next subset of customers, you can be assured of getting clean keys.

Program Dependencies. The programs used to assign production keys are listed in the Table Names matrices. Most of these programs have no dependencies (i.e., they can be executed in any order you please). The exceptions to this statement are noted in [Program Dependencies](#).

Submitting Key Assignment Programs

The key assignment programs described under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices) are submitted using a batch driver program, CIPVRNKB, and this program is executed in the staging database. You should supply the following parameters to this program:

- **Batch code.** The batch code associated with the appropriate table's key assignment program. Refer to each table listed under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices) for each key assignment batch code / program.
- **Batch thread number.** Thread number is not used and should be left blank.
- **Batch thread count.** Thread count is not used and should be left blank.
- **Batch rerun number.** Rerun number is not used and should be left blank.
- **Batch business date.** Business date is the date supplied to the key assignment programs and the date under which statistics will be logged.
- **Total number of commits.** Total number of commits is not used and should be left blank.
- **Maximum minutes between cursor re-initiation.** Maximum minutes between cursor re-initiation is not used and should be left blank.
- **User ID.** User ID is only used to log statistics for the execution of the batch job.
- **Password.** Password is not used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start (Y/N), trace program exit (Y/N), trace SQL (Y/N) and output trace (Y/N).** These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed.
- **Mode.** The proper use of this parameter will greatly speed up the key assignment step as described under [Recommendations To Speed Up Key Generation](#). This parameter has three values:
 - If you supply a mode with a value of I (initial key generation), the system allocates new keys to the rows in the staging tables (i.e., it populate the respective old key / new key table).
 - If you supply a mode with a value of D (resolve duplicate keys), the system reassigns keys that are duplicates.

- If you supply a mode with a value of B (both generate keys and resolve duplicates), the system performs both of the above steps. This is the default value if this parameter is not supplied.
- Please see [Recommendations To Speed Up Key Generation](#) for how to use this parameter to speed up the execution of these processes.

NOTE:

Parallel Key Generation. No key generation program should be run (either in mode I or B) while another program is being run unless that program is in the same tier (see [Program Dependencies](#) for a description of the tiers).

- **Start Row Number.** This parameter is only used if you are performing conversions where data already exists in the tables in the production database (subsequent conversions). In an Oracle database the key assignment routines create the initial values of keys by manipulation of the Oracle row number, starting from 1. After any conversion run, a subsequent conversion run will start with that row number again at 1, and the possibility of duplicate keys being assigned will be higher. The purpose of this parameter is to increase the value of row number by the given value, and minimize the chance of duplicate key assignment.

Recommendations To Speed Up Key Generation Programs

The following points describe ways to accelerate the execution of the key generation programs.

NOTE:

Naming convention. The convention "CK_<table_name>" is used to denote the old key / new key tables described under [The Big Picture of Key Assignment](#).

- Make the size of your rollback segments large. The exact size is dependent on the number of rows involved in your conversion. Our research has shown that processing 7 million rows generates roughly 3GB of rollback information.
 - Setup the rollback segment(s) to about 10GB with auto extend to a maximum size of 20GB to determine the high water mark
 - A next extent value on the order of 100M should be used.
- Make sure to turn off all small rollback segments (otherwise Oracle will use them rather than the large rollback segments described above).
- After the key assignment programs execute, you can reclaim the space by:
 - Keep a low value for the "minimum extent" parameter for the rollback.
 - Shrink the rollback segments and the underlying data files at the end of the large batch jobs.
- Compute statistics on the CK_<table_name> tables after every 50% increase in table size. Key generation is performed in tiers or steps because of the inheritance dependency between some tables and their keys. Although key generation for the tier currently being processed is performed by means of set-based SQL, computation of statistics between tiers will allow the database to compute the optimum access path to the keys being inherited from the **previous** tier's generation run. For Oracle users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.
- Optimal use of the **Mode** parameter under [Submitting Key Assignment Programs](#).
 - Before any key assignments, alter both the "old key" CX_ID index and the "new key" CI_ID index on the CK_<table_name> tables to be unusable.
 - Run all [key assignment tiers](#), submitting each job with MODE = "I".

- Rebuild the CX_ID and CI_ID indexes on the CK_<table_name>. Rebuilding the indexes using both the PARALLEL and NOLOGGING parameters will speed the index creation process in an Oracle DB. Statistics should be computed for these indexes.
- Run all key assignment tiers that were previously run in MODE = 'I', submitting each job with MODE = "D". This will reassign all duplicate keys.

Insert Production Data

The topics in this section describe the background processes used to populate the production database with the information in the staging database.

The Big Picture Of Insertion Programs

All insertion programs are independent and may run concurrently. Also note, all insertion programs can be run in many parallel threads as described in the next section (in order to speed execution).

Submitting Insertion Programs

The insertion programs described under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices) are submitted using a batch driver program, CIPVRNIB, and this program is executed in the staging database. You should supply the following parameters to this program:

- **Batch code.** The batch code associated with the appropriate table's insertion program. Refer to each table listed under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices) for each insertion batch code / program.
- **Batch thread number.** Thread number contains the relative thread number of the process. For example, if you want to insert accounts into production in 20 parallel threads, each of the 20 execution instances receives its relative thread number (1 through 20). Refer to [Parallel Background Processes](#) for more information.
- **Batch thread count.** Thread count contains the total number of parallel threads that have been scheduled. For example, if the account insertion process has been set up to run in 20 parallel threads, each of the 20 execution instances receives a thread count of 20. Refer to [Parallel Background Processes](#) for more information.
- **Batch rerun number.** Rerun number is not used and should be left blank.
- **Batch business date.** Business date is the date supplied to the insertion programs and the date under which statistics will be logged.
- **Total number of commits.** This is the number of commits IN TOTAL that you want to perform. For example, if you have 1,000,000 accounts and you supply a value of 100 ; then a commit will be executed for approximately every 10,000 accounts.
- **Maximum minutes between cursor re-initiation.** This should only be populated if you want to override the default value of 15.
- **User ID.** User ID is only used to log statistics for the execution of the batch job.
- **Password.** Password is not used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start (Y/N), trace program exit (Y/N), trace SQL (Y/N) and output trace (Y/N).** These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a

program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed.

Recommendations To Speed Up Insertion Programs

The following points describe ways to accelerate the execution of the insertion programs:

- Before running the first insertion program:
 - Rebuild the index on the prime key on the old key / new key table (i.e., those tables prefixed with "CK").
 - Re-analyze the statistics on the old key / new key table (i.e., those tables prefixed with "CK"). For Oracle users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.
 - Alter all indexes on the production tables being inserted into to be unusable.
- After the insertion programs have populated production data, rebuild the indexes and compute statistics for these tables. For Oracle users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.

Run Balance Control Against Production

During this step, you rerun the balance control program, but this time against production. You do this to verify the balances in production are consistent with the values of receivables converted from your legacy application.

NOTE:

Submitting this process. You submit this process in the production database. Refer to [The Big Picture of Balance Control](#) for more information about the balance control processes. Refer to [Balance Control](#) for information about the page used to view the balances generated by this process.

Validate Production

During this step, you rerun the [object validation programs](#), but this time in production. We recommend rerunning these programs to confirm that the insertion programs have executed successfully. We recommend running these programs in random sample mode (e.g., validate every 1000 th object) rather than conducting a full validation in order to save time. However, if you have time, you should run these programs in full validation mode (to validate every object). Please refer to the various "Table Names" sections above for the respective names of the programs to run.

The Validation User Interface

The topics in this section describe the various pages that assist in the conversion effort.

Validation Error Summary

Navigate to **Admin > Conversion > Validation Error Summary** to view validation errors associated with the objects defined in [Master Data](#).

Description of Page

You can use **Table Name** to restrict errors to a specific object. If this field is left blank, all errors on all objects will be displayed.

The grid contains a separate row for each type of error. The following information is displayed:

- **Table Name** is the name of the main table associated with the object.
- **Message Category** and **Message Number** define the type of error. These fields are the unique identifier of the message that describes the error (the verbiage of this message is displayed in the **Message Text** column).
- **Count** contains the number of records with this error. Press the Go To button to see the individual records with the error.

Validation Error Detail

This page is used to view validation errors of a given type associated with one of the objects defined in [Master Data](#). This transaction is not intended to be invoked from the **Admin** menu. Rather, drill into the validation details from [Validation Error Summary](#).

Description of Page

Use **Table Name**, **Message Category**, and **Message Number** to define the object and the type of error you wish to display. The grid contains a separate row for each object with the given type of error. The following information is displayed:

- **Table Name** is the name of the main table associated with the object.
- **Record Identifier** is the unique identifier of the object with the error (e.g., the person ID, the account ID, the premise ID, etc.). Press the Go To button to transfer to the maintenance page associated with the object.
- **Message Category** and **Message Number** define the type of error. These fields are the unique identifier of the message that describes the error (the verbiage of this message is displayed in the **Message Text** column).

FK Validation Summary

Navigate to **Admin > Conversion > FK Validation Summary** to view foreign key validation errors associated with the objects defined in [Master Data](#).

Description of Page

You can use **Table Name** to restrict errors to a specific object. If this field is left blank, all errors on all objects will be displayed.

The grid contains a separate row for each type of error. The following information is displayed:

- **Table Name** is the name of the main table associated with the object.
- **Count** contains the number of records on this table that have this error.
- **Foreign Key Field Names 1 to 6** contain the names of the foreign keys contained on this table that have been found to be in error.
- **Foreign Key Values 1 to 6** contain the values within the foreign key fields that are found to be in error.

FK Validation Detail

This page is used to view foreign key validation errors of a given type associated with one of the objects defined in [Master Data](#). This transaction is not intended to be invoked from the **Admin** menu. Rather, drill into the validation details from [FK Validation Summary](#).

Description of Page

Use **Table Name** to specify the table you wish to view. The names and values of the foreign key fields on the table are displayed. The grid that follows contains the primary key values of this table's records that are in error. The following information is displayed:

- **Table Name** is the name of the main table.
- **FK Fields 1 to 6** are the names of the foreign keys contained in this table. Displayed alongside the key names are the values within these fields. These identify records on other tables to which this table's record is related. For example, the CI_PREM_GEO record identified by its displayed primary keys should be related to a Premise record with the Premise ID shown - it appears in this list only if there is something amiss with this relationship.

The Staging Tables

This section describes the objects into which your legacy data is mapped. For each object, we provide the following:

- A data model.
- An indication of which tables have system-assigned keys.
- The physical table names.
- The name of the batch control to submit to validate the object.
- The name of the program (and related batch control) that validates each table for referential integrity.
- The name of the program (and related batch control) that performs key assignment for each table.
- The name of the program (and related batch control) that inserts the table's rows into production from staging.
- Suggestions to assist in the conversion process.

WARNING:

We recommend you read this document on a browser (or using Word under windows) so you can take advantage of the [Color Coding](#).

NOTE:

Column details do not appear in this document. When you're ready to examine an object's tables, use the hyperlinks in the respective Table Names section to transfer to the [data dictionary](#). The data dictionary will show you the required columns, the foreign keys (and their related tables), the source code of the program that validates the contents of the table, and a host of other information that will assist the conversion process.

WARNING:

In the data models that appear below, you will find a variety of entities that are classified as either a control table or a lookup table. Please refer to [Color Coding](#) for more information about how to recognize such an entity.

A Note About Programs in the Table Names Matrices

For each object described in the master data and transaction data sections, there is a "table names" section that includes a matrix listing the name of each table that is part of the maintenance object. Included in the matrix is information about the programs provided to perform object validation, referential integrity validation, key assignment and insertion. The following are some points about these programs:

- One object validation program exists for the entire set of tables for the maintenance object. The **Object Validation Batch Control** column indicates the batch control used to submit the object validation. Refer to [Submitting Object Validation Programs](#) for more information. Drilling down on the hypertext allows you to see more information about the batch control, including the program associated with it.
- A referential integrity validation program exists for every table whose key includes a parent key of another object. As described in [Submitting Referential Integrity Validation Programs](#), these programs are submitted using a driver supplied by the system where the batch code for the appropriate table is provided. (The driver then executes the program whose name matches the batch code). The **Referential Integrity Validation Batch Control** column indicates the table's batch control / program name.
- One key assignment program exists for the parent table for the maintenance object. As described in [Submitting Key Assignment Programs](#), these programs are submitted using a driver supplied by the system where the batch code for the appropriate table is provided. (The driver then executes the program whose name matches the batch code). The **Key Assignment Batch Control** column indicates the table's batch control / program name.
- An insertion program exists for every table for the maintenance object. As described in [Submitting Insertion Programs](#), these programs are submitted using a driver supplied by the system where the batch code for the appropriate table is provided. (The driver then executes the program whose name matches the batch code). The **Insertion Batch Control** column indicates the table's batch control / program name.

Master Data

This section describes the various "master data" objects (e.g., person, account, meter, etc.) that must be created before you can convert transaction data.

NOTE:

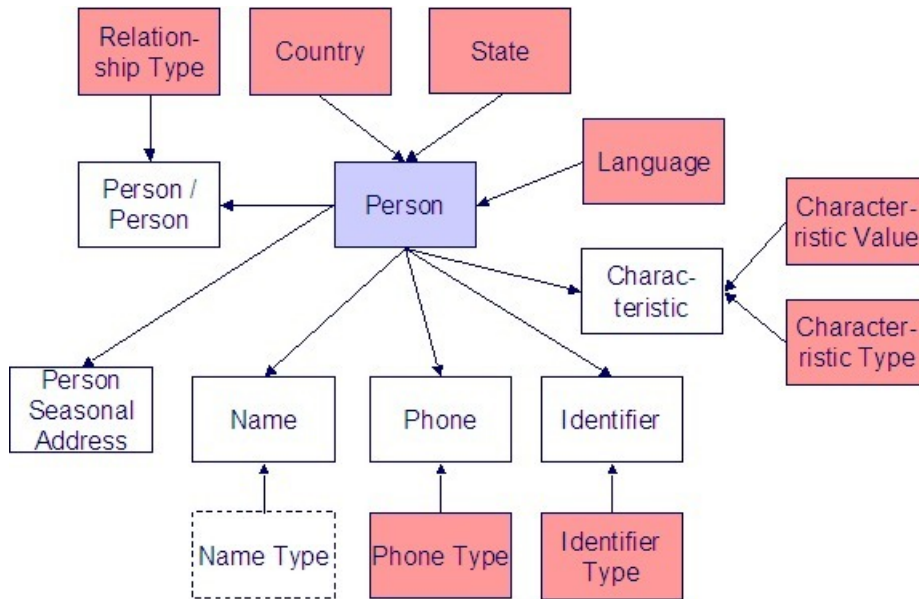
Key Assignment Dictates The Order Of Conversion. The following contents are listed in the order in which the objects should be converted in order to maintain referential integrity.

Person

Each customer must have a person and an account object. This section describes the person object. Refer to [Account](#) for details about the account object.

Person Data Model

The following data model illustrates the person object.



Person Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Person	CI_PER	Yes CI_PER_K	VAL-PER		CIPVPERK	CIPVPERI
Name	CI_PER_NAME	No. The key is PER_ID plus a sequence number.		CIPVPMNV		CIPVPMNI
Person / Person	CI_PER_PER	No. The key is PER_ID1, PER_ID2, relationship type and start date.		CIPVPEV		CIPVPEI
Phone	CI_PER_PHONE	No. The key is PER_ID plus phone type.		CIPVPPHV		CIPVPPHI
Identifier	CI_PER_ID	No. The key is PER_ID plus identifier type.		CIPVPIDV		CIPVPIDI
Characteristic	CI_PER_CHAR	No. The key is PER_ID plus an edate and a char type.		CIPVPRCV		CIPVPRCI
Seasonal Address	CI_PER_ADDR_SEAS	No. The key is PER_ID plus		CIPVPSAV		CIPVPSAI

a sequence number.

Person Suggestions

A person must have at least one row on the name table and at least one of the names must be marked as being the primary name.

A person must have at least one row on the identity table and at least one of the identities must be marked as being the primary ID.

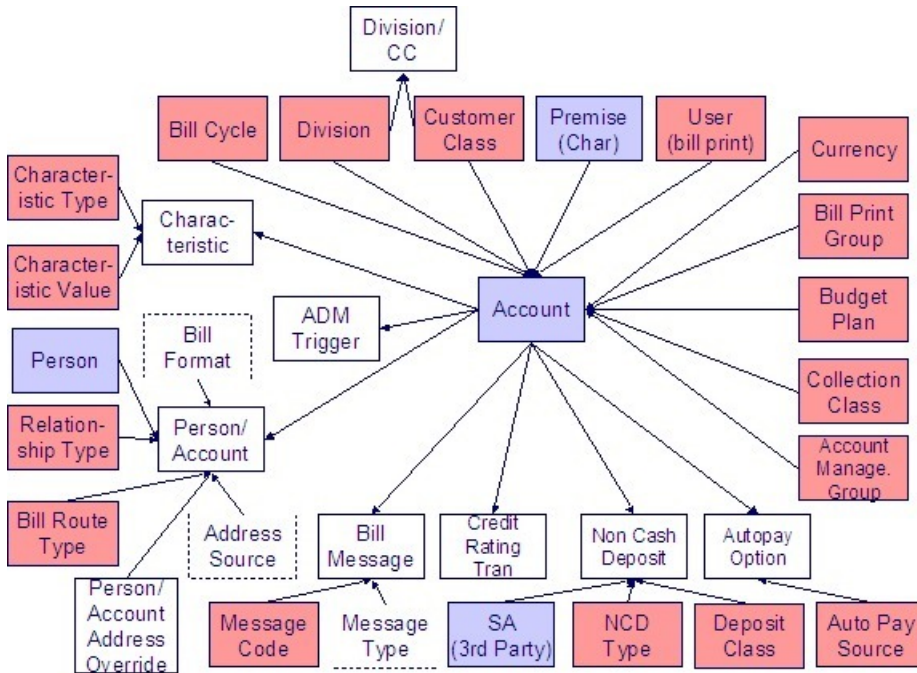
The country and state are only necessary if the person has an override mailing address.

Account

Each customer must have a person and an account object. This section describes the account object, refer to [Person](#) for details about the person object.

Account Data Model

The following data model illustrates the account object.



Account Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity	Key Assignment Batch Control	Insertion Batch Control
-----------------	------------	----------------	---------------------------------	-----------------------	------------------------------	-------------------------

			Validation Batch Control		
Account	CI_ACCT	Yes CI_ACCT_K VAL-ACCT		CIPVACCK	CIPVACCI
Bill Message	CI_ACCT_MSG	No. The key is account ID plus bill message code.		CIPVMSGV	CIPVMSGI
Autopay Option	CI_ACCT_APAY	Yes CI_ACCT_APAY_K		CIPVAAPV	CIPVAAPK Has dependencies
Characteristic	CI_ACCT_CHAR	No. The key is ACCT_ID plus an edate and a char type.		CIPVACHV	CIPVACHI
Person/Account	CI_ACCT_PER	No. The key is account ID plus person ID.		CIPVACPV	CIPVACPI
Person/Account Address Override	CI_PER_ADDR_OVRD	No. The key is Account ID plus Person ID		CIPVPAOV	CIPVPAOI
Non Cash Deposit	CI_NCD	No. The key is account ID plus seq number		CIPVNCDV	CIPVNCDI
Credit Rating Tran	CI_CR_RAT_HIST	Yes CI_CR_RAT_HIST_K		CIPVCRTV	CIPVCRRK Has dependencies
ADM Trigger	CI_ADM_RVW_SCH	No. The key is account ID plus date		CIPVARSV	CIPVARSI

Account Suggestions

An account must have at least one row on the account / person table and at least one account / person must be marked as being the main customer. Please see column notes for the account / person table for inter-field validation in respect of the various switches (e.g., if main customer switch is on, then the person must also be financially responsible).

We recommend storing an ADM trigger ([CI_ADM_RVW_SCH](#)) for every account where the trigger date is the conversion date. This will cause the account to be reviewed by the [account debt monitor](#) when it next runs. We have supplied a dedicated batch process for this purpose that simply inserts a row in this table with the review date set equal to the current date. This will ensure that all converted accounts are reviewed after they are inserted into production. This program is named CIPVADMB and goes by the batch control ID of CNV-ADM.

If your legacy system has the equivalent of a credit rating or a cash only score, you should create credit rating transactions. The values you create need to be consistent with the base and threshold credit rating and cash only points on the installation record. Refer to the business process guide - customer information - how are credit rating transactions created for more information.

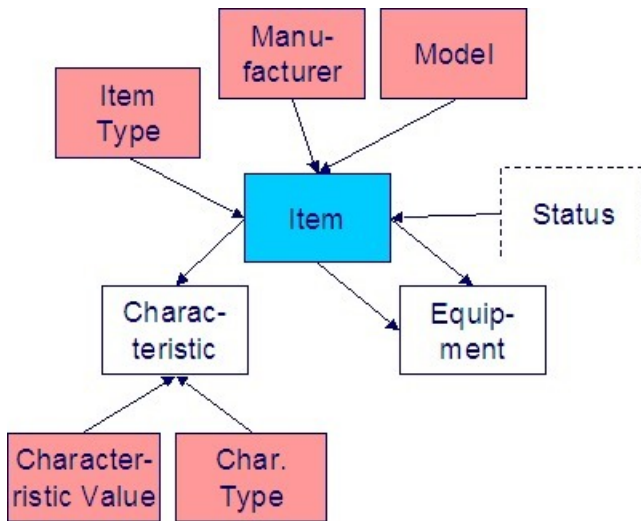
Item

Each badged piece of equipment must have an item. Examples of items include badged street lamps, current transforms, backflow devices, voltage regulators, etc.

If the item is currently installed at a service point, you must link the item to the service point by inserting a row on the [SP Item History and On Off Event](#) tables.

Item Data Model

The following data model illustrates the item object.



Item Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Item	CI_ITEM	Yes CI_ITEM_K	VAL-ITEM		CIPVITMK	CIPVITMI
Characteristic	CI_ITEM_CHAR	No. The key is ACCT_ID plus an edate and a char type.		CIPVITCV		CIPVITCI
Equipment	CI_ITEM_EQ	No. The key is item ID plus equipment (item) ID.		CIPVIEQV		CIPVIEQI

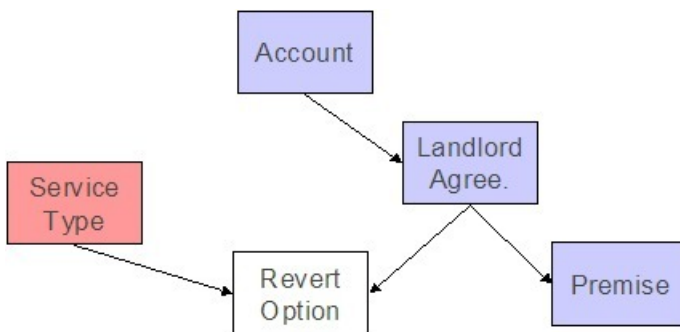
Item Suggestions

N/A.

Landlord

Landlord Data Model

The following data model illustrates the landlord object.



Landlord Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Landlord	CI_LANDLORD	Yes CI_LANDLORD_K	VAL-LL		CIPVLNDK	CIPVLNDI
Revert Option	CI_LL_DETAIL	No. The key is LL_ID plus service type		CIPVLLDV		CIPVLLDI

Landlord Suggestions

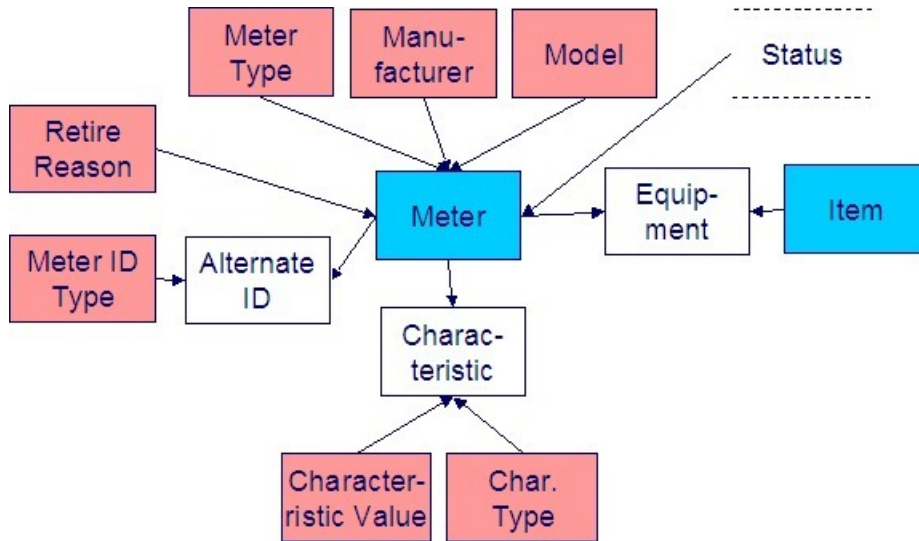
N/A.

Meter

Every meter must have a meter object and every meter object must have a meter configuration. This section describes the meter object. Refer to [Meter Configuration](#) for information about the meter configuration object.

Meter Data Model

The following data model illustrates the meter object.



Meter Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Meter	CI_MTR	Yes CI_MTR_K	VAL-MTR		CIPVMTRK Has dependencies	CIPVMTRI
Characteristic	CI_MTR_CHAR	No. The key is MTR_ID plus an edate and a char type.		CIPVMTCV		CIPVMTCI
Equipment	CI_MTR_EQ	No. The key is meter ID plus equipment (meter) ID.		CIPVMEQV		CIPVMEQI
Alternate ID	CI_MTR_ID	No. The key is MTR_ID plus meter id type.		CIPVMIDV		CIPVMIDI

Meter Suggestions

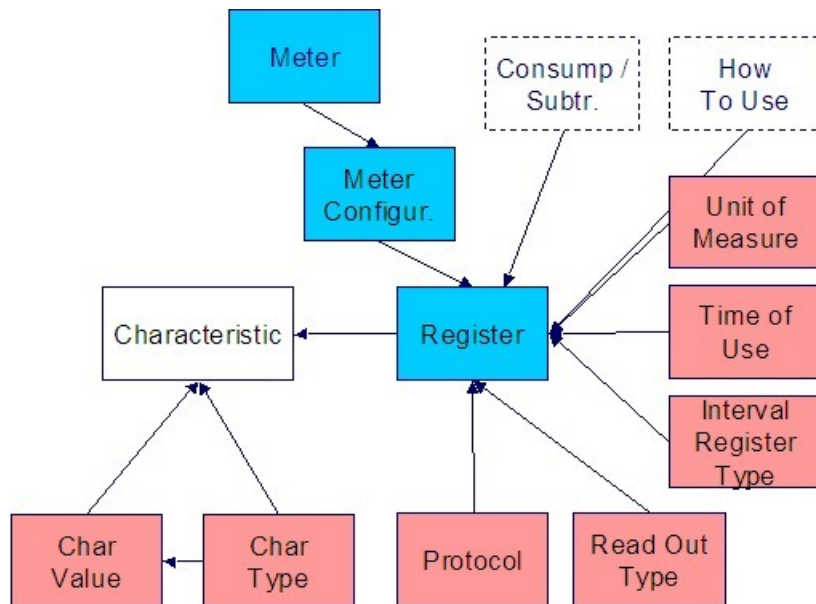
N/A.

Meter Configuration

Every meter must have a meter object and every meter object must have a meter configuration. This section describes the meter configuration object. Refer to [Meter](#) for information about the meter object.

Meter Configuration Data Model

The following data model illustrates the meter configuration object.



Meter Configuration Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Meter Configuration	CI_MTR_CONFIG	Yes CI_MTR_CONFIG_K	VAL-CFG		CIPVMTGK Has dependencies	CIPVMTGI
Register	CI_REG	Yes CI_REG_K		CIPREGV	CIPREGK Has dependencies	CIPREGI

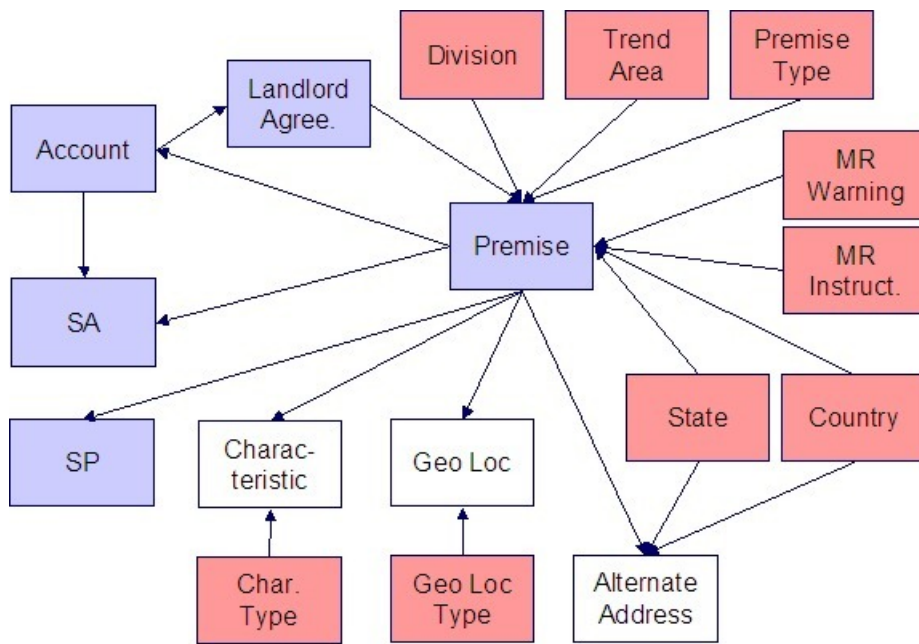
Meter Configuration Suggestions

N/A.

Premise

Premise Data Model

The following data model illustrates the premise object.



Premise Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Premise	CI_PREM	Yes CI_PREM_K	VAL-PREM		CIPVPRMK	CIPVPRMI

[Has dependencies](#)

Characteristic	CI_PREM_CHAR	No. The key is PREM_ID plus an edate and a char type.	CIPVPCHV	CIPVPCHI
Geo Loc	CI_PREM_GEO	No. The key is PREM_ID plus geo loc type.	CIPVPGOV	CIPVPGOI
Alternate Address	CI_PRM_ALT_ADDR	Yes CI_PRM_ALT_ADDR_K	CIPVAPAV	CIPVAPAK CIPVAPAI

[Has dependencies](#)

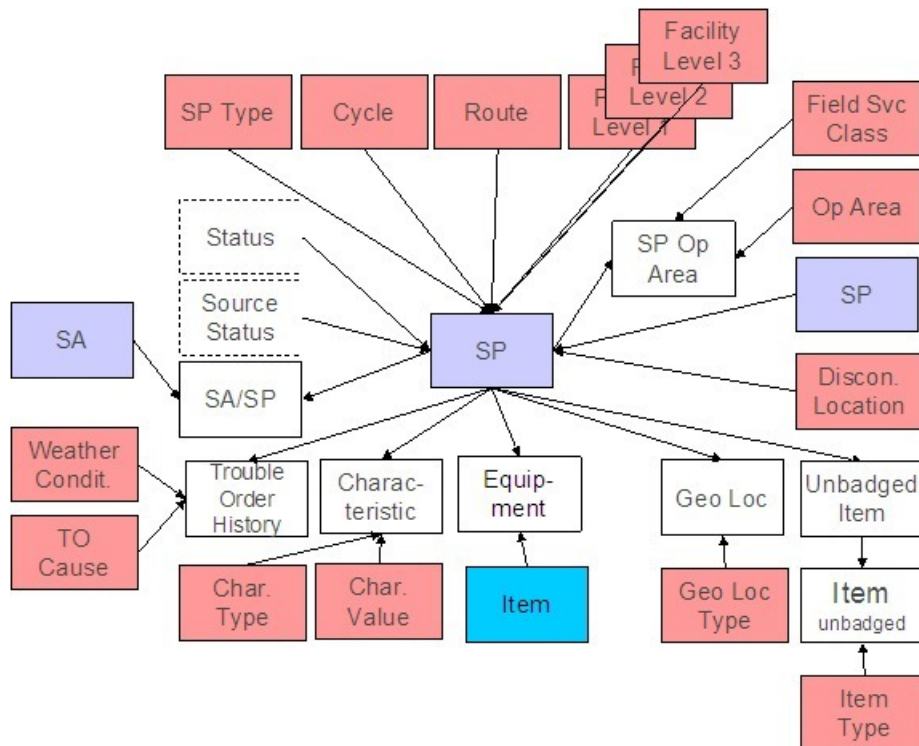
Premise Suggestions

N/A.

Service Point

Service Point Data Model

The following data model illustrates the service point object.



Service Point Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Service Point	CI_SP	Yes CI_SP_K	VAL-SP		CIPVSPPK Has dependencies	CIPVSPPI
Characteristic	CI_SP_CHAR	No. The key is SP_ID plus an edate and a char type.		CIPVSPCV		CIPVSPCI
Equipment	CI_SP_EQ	No. The key is service point ID plus equipment (service point) ID.		CIPVSEQV		CIPVSEKI
Geo Loc	CI_SP_GEO	No. The key is service point ID plus geo type.		CIPVSPGV		CIPVSPGI
Unbadged Item	CI_SP_MULT_ITEM	No. The key is SP_ID plus edate.		CIPVSPMV		CIPVSPMI
Item	CI_MULT_ITEM	No. The key is SP_ID, edate and item type.		CIPVSMIV		CIPVSMII
SP Op Area	CI_SP_OP_AREA	No. The key is SP_ID plus field service classification code		CIPVSPOV		CIPVSPOI
SA/SP	CI_SA_SP (note, this table really belongs to the SA object, it is included here for completeness)	Yes CI_SA_SP_K	CIPVSVAB	CIPVSAPV	CIPVSPPK Has dependencies	CIPVSAPI

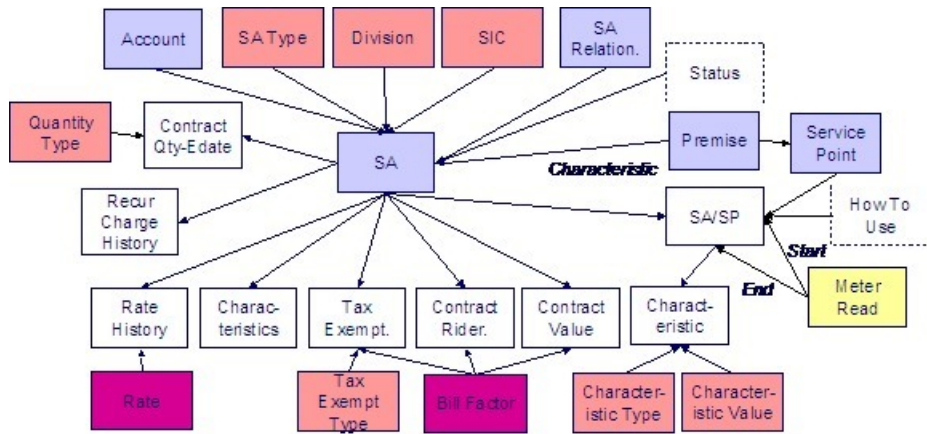
Service Point Suggestions

N/A.

Service Agreement

Service Agreement Data Model

The following data model illustrates the service agreement object.



Service Agreement Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Service Agreement	CI_SA	Yes CI_SA_K	VAL-SA		CIPVSAK	CIPVSAI
Characteristic	CI_SA_CHAR	No. The key is SA_ID plus an edate and a char type.		CIPVSACV		CIPVSACI
Contract Quantity Edate	CI_SA_CONT_QTY	No. The key is service agreement ID plus quantity type plus edate.		CIPVSAQV		CIPVSAQI
Message	CI_SA_MSG	No. The key is service agreement ID plus Bill message code.		CIPVSMGV		CIPVSMGI
Recurring Charge	CI_SA_RCHG_HIST	No. The key is service agreement ID plus edate.		CIPVSARV		CIPVSARI
SA Relationship	CI_SA_REL	Yes CI_SA_REL_K	VAL-SARL	CIPVSRLV	CIPVSRLK	CIPVSRLI

Has
dependencies

Rate History	CI_SA_RS_HIST	No. The key is service agreement ID plus edate.	CIPVSAHV		CIPVSAHI
SA/SP	CI_SA_SP	Yes CI_SA_SP_K	CIPVSAPV	CIPVSSPK	CIPVSAPI
SA/SP Characteristic	CI_SA_SP_CHAR	No. The key is SA/SP Id plus char type plus effective date.	CIPVSSCV		CIPVSSCI
Tax Exempt	CI_SA_CONTERM - this table is also used for the next 2 entities, the key contains CONTERM_TYPE_FLG that controls the entity	No. This key is service agreement ID plus CONTERM_TYPE_FLG plus BF_CD plus START_DT	CIPVSAOV		CIPVSAOI
Contract Rider	CI_SA_CONTERM - this table is also used for the previous and next entities, the key contains CONTERM_TYPE_FLG that controls the entity	No. This key is service agreement ID plus CONTERM_TYPE_FLG plus BF_CD plus START_DT	CIPVSAOV		CIPVSAOI
Contract Value	CI_SA_CONTERM - this table is also used for the previous 2 entities, the key contains CONTERM_TYPE_FLG that controls the entity	No. This key is service agreement ID plus CONTERM_TYPE_FLG plus BF_CD plus START_DT	CIPVSAOV		CIPVSAOI

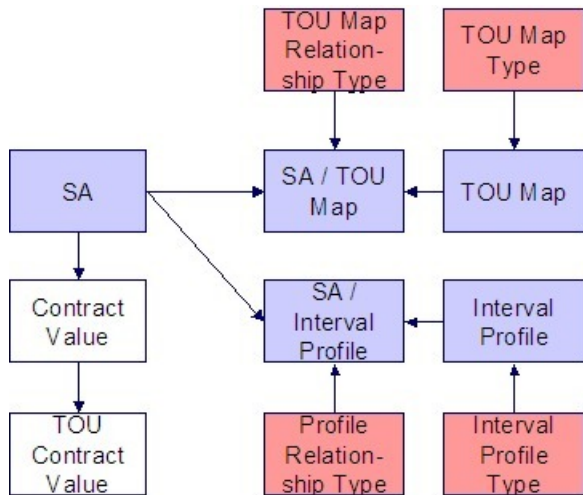
Service Agreement Suggestions

N/A.

SA Interval Billing

SA Interval Billing Data Model

The following data model illustrates the Interval Billing objects related to the service agreement.



SA Interval Billing Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
TOU Map	CI_TOU_MAP	Yes CI_TOU_MAP_K	VAL-TMAP		CIPVTMAK Has dependencies	CIPVTMAI
TOU Map Lang	CI_TOU_MAP_L	No. The key is TOU_MAP_ID plus language code.		CIPVTMLV		CIPVTMLI
Interval Profile	CI_INTV_PF	Yes CI_INTV_PF_K	VAL-INPF		CIPVINPK Has dependencies	CIPVINPI
Interval Profile Lang	CI_INTV_PF_L	No. The key is INTV_PF_ID plus language code		CIPVINLV		CIPVINLI
SA / Interval Profile	CI_SA_INTV_PF	No. The key is service agreement ID plus INTV_PF_REL_TYP_CD plus START_DTTM plus INTV_PF_ID		CIPVSIFV		CIPVSIPI

SA / TOU Map	CI_SA_TOU_MAP	No. The key is service agreement ID plus TMAP_REL_TYPE_CD plus START_DTTM plus TOU_MAP_ID	CIPVSTMV	CIPVSTMI
TOU Contract Value	CI_TOU_CONT_VAL	No. The key is service agreement ID plus CONTERM_TYPE_FLG plus START_DT plus BF_CD plus TOU_GRP_CD plus TOU_CD	CIPVTCVV	CIPVTCVI

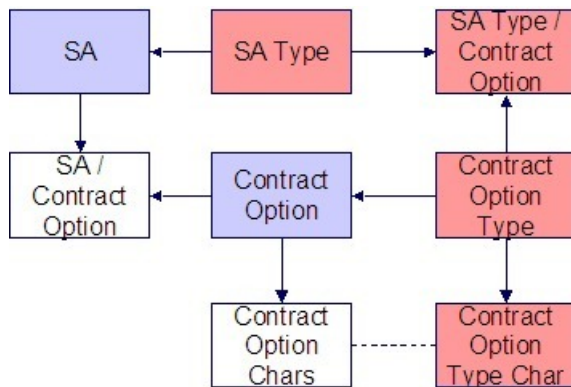
SA Interval Billing Suggestions

N/A.

Contract Options

Contract Options Data Model

The following data model illustrates the contract options objects.



Contract Options Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity	Key Assignment Batch Control	Insertion Batch Control
-----------------	------------	----------------	---------------------------------	-----------------------	------------------------------	-------------------------

			Validation Batch Control		
Contract Option	CI_COP	Yes CI_COP_K	VAL-COP	CIPVCOPK	CIPVCOPI
				Has dependencies	
Contract Option Language	CI_COP_L	No. The key is CONT_OPT_ID plus language code		CIPVCOLV	CIPVCOLI
Contract Option Characteristics	CI_COP_CHAR	No. The key is CONT_OPT_ID plus CHAR_TYPE_CD plus EFFDT		CIPVCCFV	CIPVCCAI
SA / Contract Option	CI_SA_COP	Yes CI_SA_COP_K		CIPVSCPV	CIPVSCP
				Has dependencies	CIPVSCPI

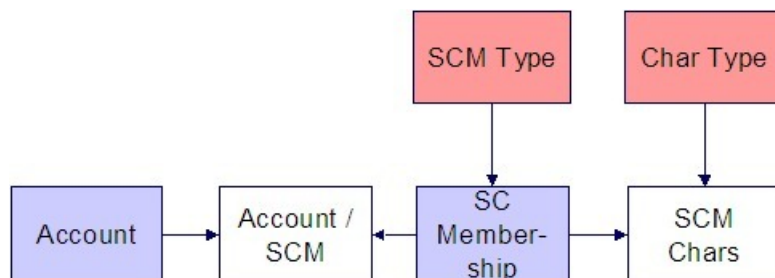
Contract Options Suggestions

N/A.

Service Credit Membership

Service Credit Membership Data Model

The following data model illustrates the Service Credit Membership objects.



Service Credit Membership Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control

Service Credit Membership	CI_SCM	Yes CI_SCM_K	VAL-SCM	CIPVSCBV	CIPVSCMK	CIPVSCMI
Service Credit Membership / Account	CI_SCM_ACCT	No. The key is SCM_ID plus ACCT_ID.		CIPVSCAV		CIPVSCAI
Service Credit Membership Characteristics	CI_SCM_CHAR	No. The key is SCM_ID plus CHAR_TYPE_CD plus EFFDT.		CIPVSCCV		CIPVSCCI

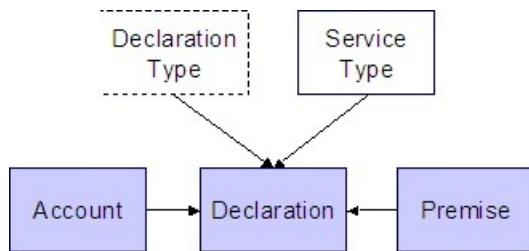
Service Credit Membership Suggestions

N/A.

Declaration

Declaration Data Model

The following data model illustrates the Declaration object.



Declaration Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Declaration	CI_DCL	Yes CI_DCL_K	VAL-DCL		CIPVDCRK	CIPVDCRI

Declaration Suggestions

N/A.

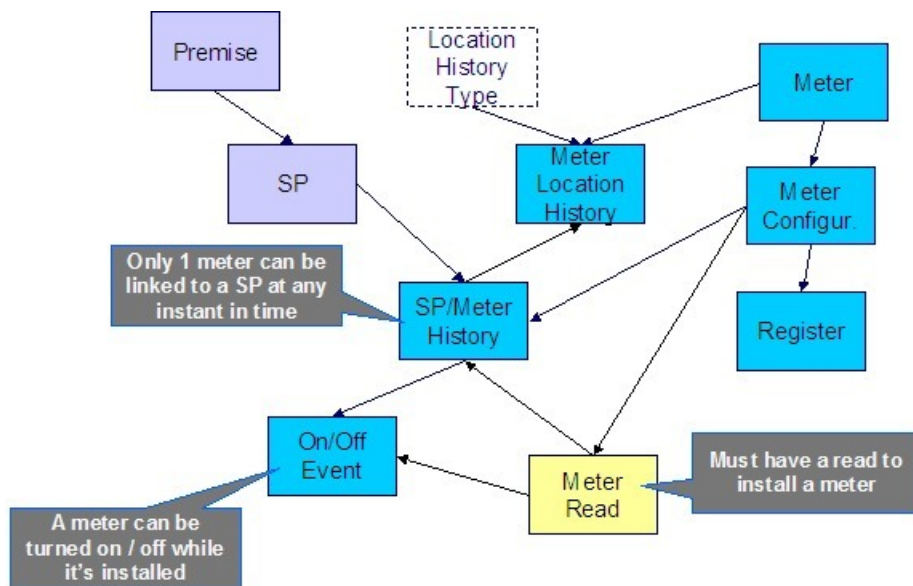
Transaction Data

This section describes the tables in which your transaction data (e.g., bills, payments, meter reads, customer contacts, etc.) resides.

SP / Meter History and On/Off Event

SP / Meter Data Model

The following data model illustrates the service point / meter installation object.



SP / Meter Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
SP/Meter History	CI_SP_MTR_HIST	Yes CI_SP_MTR_HIST_K	CIPVSMHV	CIPVSMHK Has dependencies	CIPVSMHI
On/Off Event	CI_SP_MTR_EVT	No. The key is meter history ID plus sequence number.	CIPVSMEV		CIPVSMEI
Meter Location History	CI_MTR_LOC_HIS	Yes. CI_MTR_LOC_HIS_K	CIPVMLHV	CIPVMLHK	CIPVMLHI

SP / Meter Suggestions

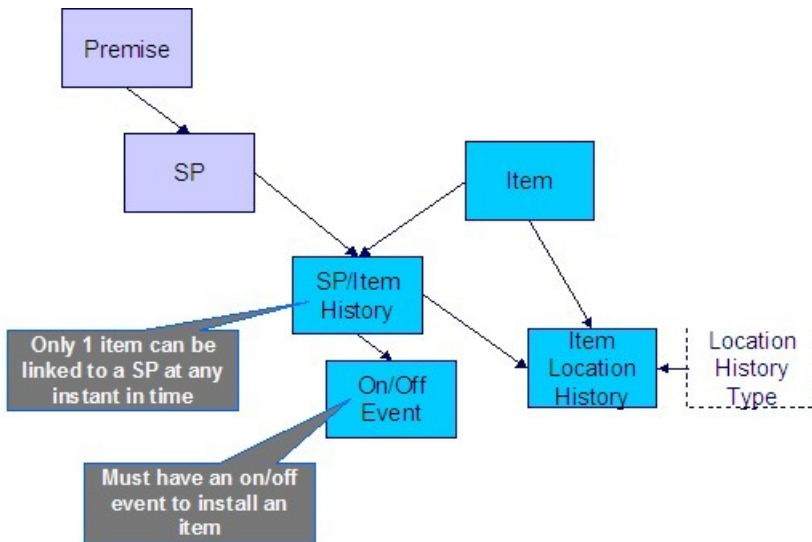
In order to link a meter to a service point, you must

- Link the meter's meter configuration to the service point by inserting a record on the CI_SP_MTR_HIST table.
- In addition, you must also create a CI_SP_MTR_EVT record. Note the following about this record:
 - The value of the SP_MTR_EVT_FLG should be I (for installation).
 - The value of the MTR_ON_OFF_FLG should be 1 (on).
 - It must reference a read whose read date is the installation date. The reading can be a dummy value unless this customer has not been billed since the install date (i.e., the installation has taken place recently). In this situation, this read must be the true start read for the customer. Note, this read should also be linked to the SA/SP record associated with the SA that's linked to the SP as its start read.

SP / Item History and On/Off Event

SP / Item Data Model

The following data model illustrates the service point / item installation object.



SP / Item Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
SP/Item History	CI_SP_ITEM_HIST	Yes CI_SP_ITEM_HIST_K	CIPVSIHV	CIPVSIHK Has dependencies	CIPVSIHI

On/Off Event	CI_SP_ITEM_EVT	No. The key is meter history ID plus sequence no.	CIPVSIEV		CIPVSIEI
Item Location History	CI_ITEM_LOC_HIS	Yes. CI_ITEM_LOC_HIS_K	CIPVILHV	CIPVILHK	CIPVILHI

SP / Item Suggestions

In order to link an item to a service point, you must

- Link the item to the service point by inserting a record on the CI_SP_ITEM_HIST table.
- In addition, you must also create a CI_SP_ITEM_EVT record. Note the following about this record:
 - The value of the SP_ITEM_EVT_FLG should be I (for installation).
 - The value of the ITEM_ON_OFF_FLG should be 1 (on).

Bill

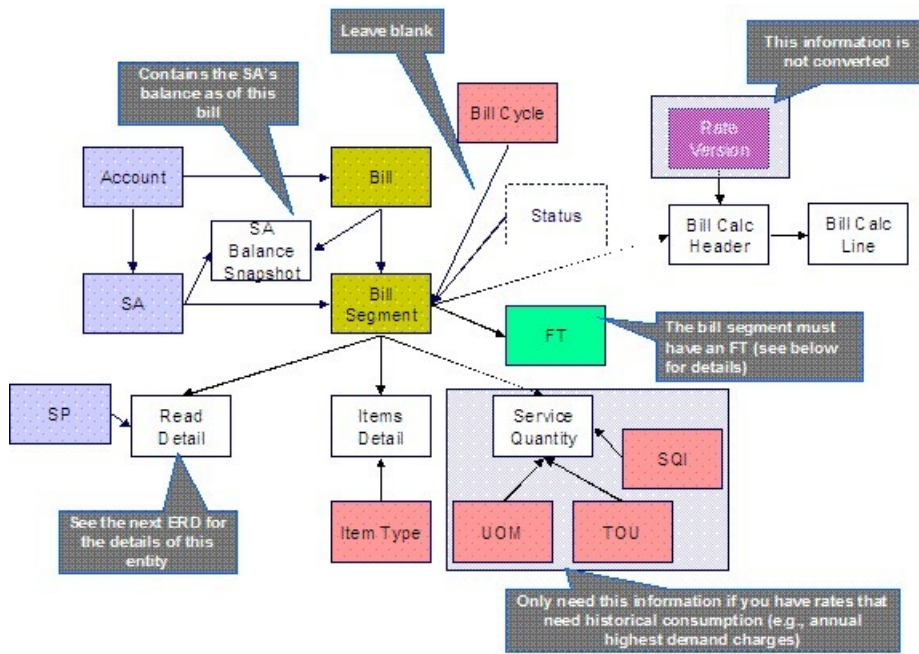
Bill Data Model

This section contains information about the Bill's data model:

- Bill - Main
- Bill - Read Details
- Bill - FT
- Bill Characteristics
- Bill Messages, Bill Routing, and Bill Review Schedule

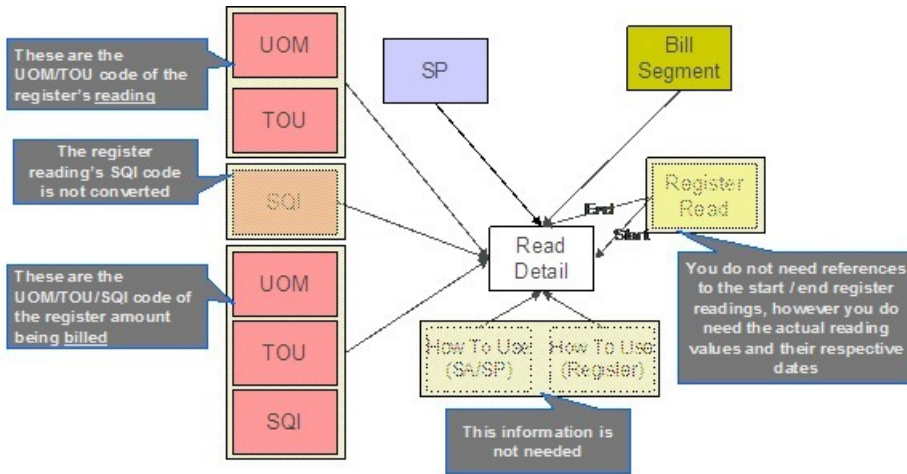
Bill - Main

The following data model illustrates the bill object.



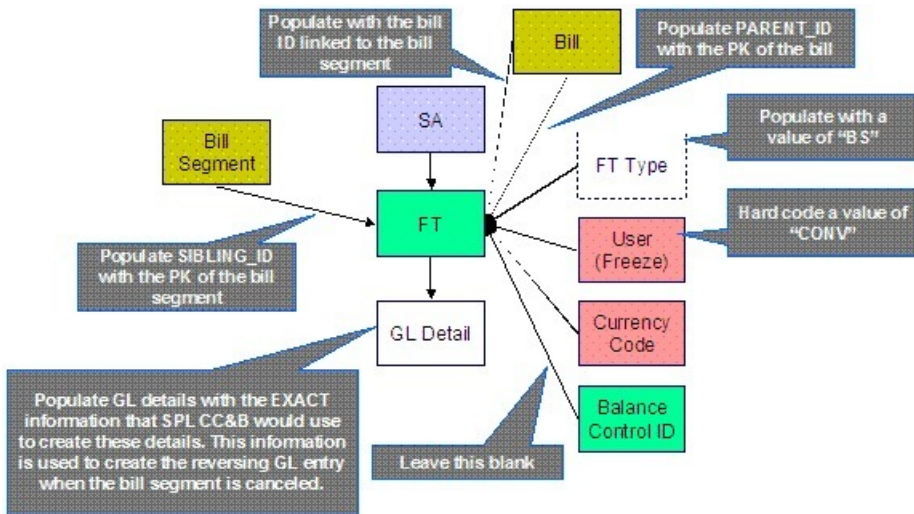
Bill - Read Details

The following data model illustrates the FK references on the read detail entity (a bill segment has one or more read details if the bill segment is associated with metered service).



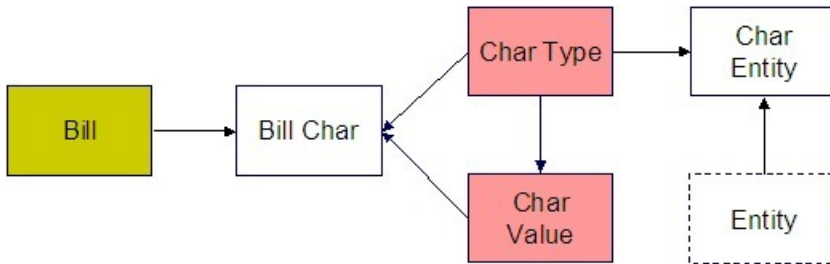
Bill - FT

The following data model illustrates the FT that must be associated with a bill segment.



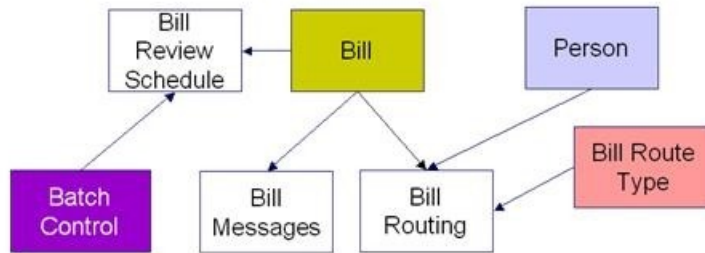
Bill Characteristics

The following data model illustrates Bill Characteristics.



Bill Messages, Bill Routing, and Bill Review Schedule

The following data model illustrates Bill Messages, Bill Routing, and Bill Review Schedule.



Bill Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Bill	CI_BILL	Yes CI_BILL_K	CIPVBLLV	CIPVBILK Has dependencies	CIPVBLLI
SA Balance Snapshot	CI_BILL_SA	No. The key is bill ID plus SA ID.	CIPVBSAV		CIPVBSAI

Bill Segment	CI_BSEG	Yes CI_BSEG_K	CIPVSEGV	CIPVBSGK Has dependencies	CIPVSEGI
Calc Header	CI_BSEG_CALC	No. The key is bill segment ID and a sequence number	CIPVBSCV		CIPVBSCI
Calc Lines	CI_BSEG_CALC_LN	No. The key is bill segment ID, the header sequence number, and a sequence number	CIPVBSLV		CIPVBSLI
Read Detail	CI_BSEG_READ	No. The key is bill segment ID, SP ID and a sequence number	CIPVSRRV		CIPVSRRI
Item Detail	CI_BSEG_ITEM	No. The key is bill segment id and a sequence number	CIPVBSIV		CIPVBSII
Service Quantity	CI_BSEG_SQ	No. The key is bill segment ID, uom code, tou code and SQI code	CIPVSQTV		CIPVSQTI
FT (financial transaction)	CI_FT	Yes CI_FT_K	CIPVFTFV	CIPVFTXK Has dependencies	CIPVFTFI
FT GL (FT general ledger)	CI_FT_GL	No. The key is FT ID and a GL sequence number	CIPVFTGV		CIPVFTGI
Characteristics	CI_BILL_CHAR	No. The key is bill ID, char type code and a sequence number	CIPVBCHV		CIPVBCHI
Bill Messages	CI_BILL_MSGS	No. The key is bill ID and bill message code.	CIPVBLMV		CIPVBLMI
Bill Routing	CI_BILL_ROUTING	No. The key is bill ID and a sequence number	CIPVBLRV		CIPVBLRI
Bill Review Schedule	CI_BILL_RVW_SCH	No. The key is bill ID, bill review date and batch code.	CIPVBRVV		CIPVBRVI

Bill Suggestions

Most companies have found it impossible to load bill segment item, bill calculation header and lines with sufficient information and therefore these tables are not populated. See the comments in the above ERD's for more information.

Please populate the columns on the FT that's associated with the bill segment as follows:

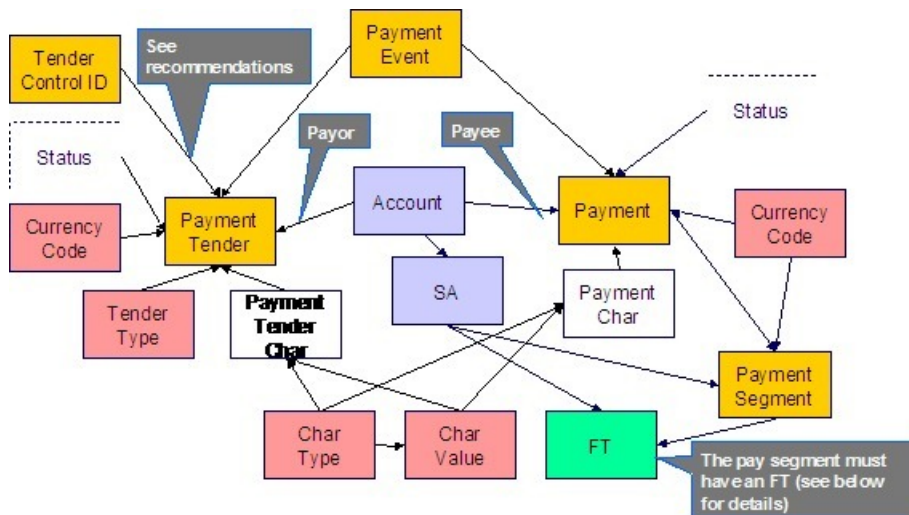
- CUR_AMT should be set equal to the bill segment amount

- PAY_AMT should be set equal to the bill segment amount
- CRE_DTTM should be set equal to the bill segment end date / time
- FREEZE_SW should be "Y"
- FREEZE_DTTM should be set equal to the bill segment end date / time
- ARS_DT should be set equal to the bill segment end date
- CORRECTION_SW should be "N"
- REDUNDANT_SW should be "N"
- NEW_DEBIT_SW should be "N"
- NOT_IN_ARS_SW should be set to "N"
- SHOW_ON_BILL_SW should be set to "N"
- ACCOUNTING_DT should be set to the current date
- SCHED_DISTRIB_DT should be left blank
- CURRENCY_CD should be the currency on the installation record
- BAL_CTL_GRP_ID should be left blank
- XFERRED_OUT_SW should be set to "Y"
- PARENT_ID should be set to the bill ID
- SIBLING_ID should be set to the bill segment ID
- Do NOT create any GL details for the FT. If GL details are converted, ensure they are populated with the EXACT information SPL CC&B would use to create them. This information is used to create the reversing GL entry when the bill segment is canceled.

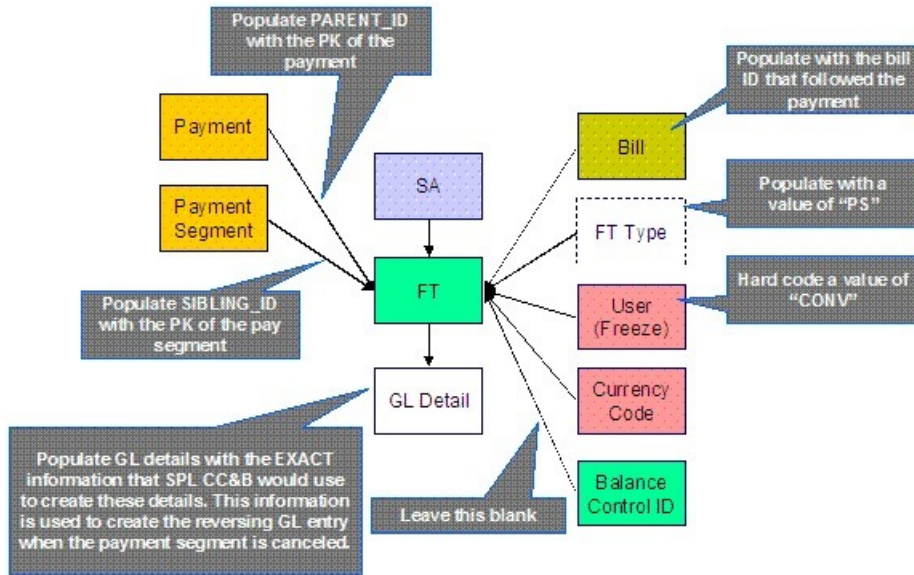
Payment

Payment Data Model

The following data model illustrates the payment object.



The following data model illustrates the FT that must be associated with a payment segment.



Payment Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Payment	CI_PAY	Yes CI_PAY_K	CIPVPAYV	CIPVPAYK Has dependencies	CIPVPAYI
Payment Event	CI_PAY_EVENT	Yes CI_PAY_ EVENT_K		CIPVPYEK Has dependencies	CIPVPYEI
Payment Event Characteristic	CI_PAY_EVT_CHAR	No. The key is PAY_ EVENT_ID and a char type.	CIPVBLCV		CIPVBLCI
Payment Tender	CI_PAY_TNDR	Yes CI_PAY_TNDR_ K	CIPVTNDV	CIPVTNDK Has dependencies	CIPVTNDI
Payment Segment	CI_PAY_SEG	Yes CI_PAY_SEG_K	CIPVPSGV	CIPVPSGK Has dependencies	CIPVPSGI
FT (financial transaction)	CI_FT	Yes CI_FT_K	CIPVFTFV	CIPVFTXK Has dependencies	CIPVFTFI
FT GL (FT general ledger)	CI_FT_GL	No. The key is FT id and a GL sequence number	CIPVFTGV		CIPVFTGI
Payment Tender Characteristic	CI_PAY_TNDR_ CHAR	No. The key is PAY_ TENDER_ID, plus a sequence number and a char type	CIPVTNCV		CIPVTNCI

Payment Characteristic	CI_PAY_CHAR	No. The key is PAY_ ID, plus a sequence number and a char type	CIPVPYCV	CIPVPYCI
---------------------------	-------------	---	----------	----------

Payment Suggestions

We recommend that you use the system to create a single deposit control and link to it a single tender control using the PRODUCTION tables. The tender control should reference a tender source of "conversion". Use the prime key of the tender control as the foreign key on the tenders that you insert into the STAGING tables. This means you will have an invalid foreign key relationship on CI_PAY_TNDR (it will reference a tender control that doesn't exist).

After converting the payments:

- Re-access the tender control in production and enter the appropriate amounts (per tender type) to balance the tender control.
- Re-access the deposit control in production and enter the appropriate amounts to balance the deposit control.

Please populate the columns on the FT that's associated with the payment segment as follows:

- CUR_AMT should be set equal to the payment segment amount
- PAY_AMT should be set equal to the payment segment amount
- CRE_DTTM should be set equal to the payment segment date / time
- FREEZE_SW should be "Y"
- FREEZE_DTTM should be set equal to the payment segment date / time
- ARS_DT should be set equal to the payment segment date
- CORRECTION_SW should be "N"
- REDUNDANT_SW should be "N"
- NEW_DEBIT_SW should be "N"
- NOT_IN_ARS_SW should be set to "N"
- SHOW_ON_BILL_SW should be set to "N" on all payments other than payments that have been received since the last bill. For recent payments that you want to show on the next bill, this switch must be "Y"
- ACCOUNTING_DT should be set to the current date
- SCHED_DISTRIB_DT should be left blank
- CURRENCY_CD should be the currency on the installation record
- BAL_CTL_GRP_ID should be left blank
- XFERRED_OUT_SW should be set to "Y"
- PARENT_ID should be set to the payment ID
- SIBLING_ID should be set to the payment segment ID
- Do NOT create any GL details for the FT. If GL details are converted, ensure they are populated with the EXACT information SPL CC&B would use to create them. This information is used to create the reversing GL entry when the payment segment is canceled.

Adjustment

Adjustment Data Model

Adjustment Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Adjustment	CI_ADJ	Yes CI_ADJ_K	CIPVADJV	CIPVADJK Has dependencies	CIPVADJI
Adjustment Characteristic	CI_ADJ_CHAR	No. The key is ADJ_ ID and a char type.	CIPVADCV		CIPVADCI
Adjustment A/P Request	CI_ADJ_APREQ	Yes CI_ADJ_ APREQ_K	CIPVAPRV	CIPVAPRK Has dependencies	CIPVAPRI
FT (financial transaction)	CI_FT	Yes CI_FT_K	CIPVFTFV	CIPVFTXK Has dependencies	CIPVFTFI
FT GL (FT general ledger)	CI_FT_GL	No. The key is FT ID and a GL sequence number	CIPVFTGV		CIPVFTGI
FT Process	CI_FT_PROC	No. The key is FT id and a sequence number	CIPVFTPV		CIPVFTPI

Adjustment Suggestions

Please populate the columns on the FT that's associated with the adjustment as follows:

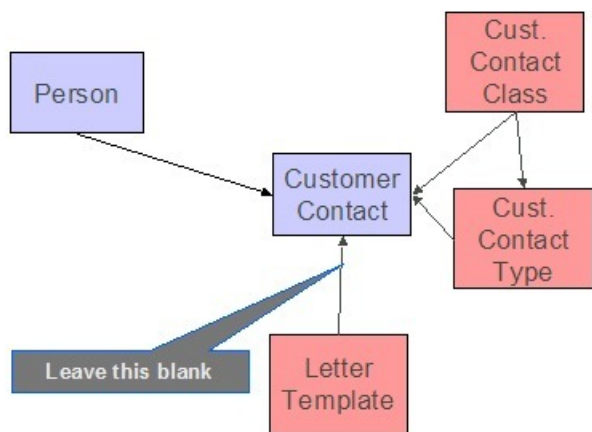
- CUR_AMT should be set equal to the adjustment amount
- PAY_AMT should be set equal to the adjustment amount
- CRE_DTTM should be set equal to the adjustment date / time
- FREEZE_SW should be "Y"
- FREEZE_DTTM should be set equal to the adjustment date / time
- ARS_DT should be set equal to the adjustment date
- CORRECTION_SW should be "N"
- REDUNDANT_SW should be "N"
- NEW_DEBIT_SW should be "N"

- NOT_IN_ARS_SW should be set to "N"
- SHOW_ON_BILL_SW should be set to "N" on all adjustments other than adjustments that have been generated since the last bill. For recent adjustments that you want to show on the next bill, this switch must be "Y"
- ACCOUNTING_DT should be set to the current date
- SCHED_DISTRIIB_DT should be left blank
- CURRENCY_CD should be the currency on the installation record
- BAL_CTL_GRP_ID should be left blank
- XFERRED_OUT_SW should be set to "Y"
- PARENT_ID should be set to the adjustment's adjustment type
- SIBLING_ID should be set to the adjustment ID
- Do NOT create any GL details for the FT. If GL details are converted, ensure they are populated with the EXACT information SPL CC&B would use to create them. This information is used to create the reversing GL entry when the adjustment is canceled.

Customer Contact

Customer Contact Data Model

The following data model illustrates the Customer Contact object.



Customer Contact Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Customer Contact	CI_CC	Yes CI_CC_K	CIPVCSCV	CIPVCCTK Has dependencies	CIPVCSCI

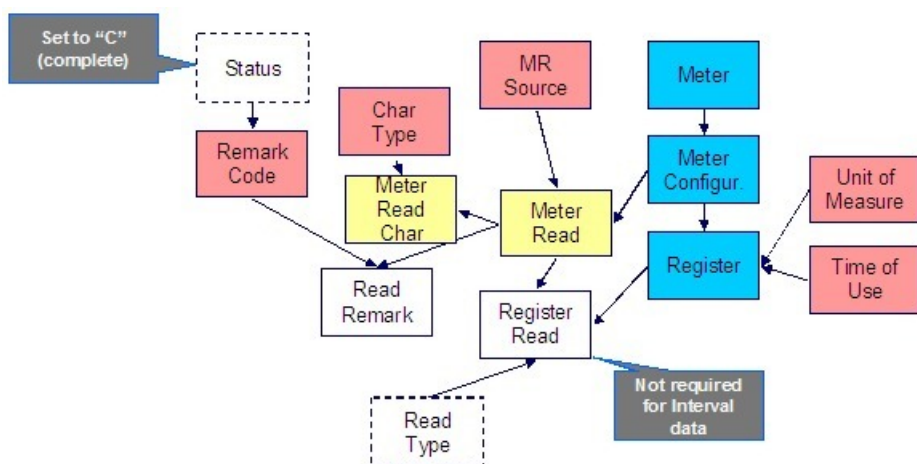
Customer Contact Suggestions

N/A

Meter Read

Meter Read Data Model

The following data model illustrates the meter read object.



Meter Read Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Meter Read	CI_MR	Yes CI_MR_K	CIPVMRDV	CIPVMRDK Has dependencies	CIPVMRDI
Register Read	CI_REG_READ	Yes CI_REG_READ_K	CIPVRGRV	CIPVRRDK Has dependencies	CIPVRGRI
Read Remark	CI_MR_REM	No. Key is meter read plus read ID.	CIPVMRMV		CIPVMRMI
Meter Read Characteristics	CI_MR_CHAR	No. The key is MR_ID plus a sequence number and a char type.	CIPVMRCV		CIPVMRCI

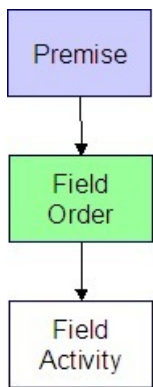
Meter Read Suggestions

N/A

Field Order

Field Order Data Model

The following data model illustrates the Field Order object.



Field Order Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Field Order	CI_FO	Yes CI_FO_K	VAL-FO	CIPVFORV	CIPVFORK Has dependencies	CIPVFORI

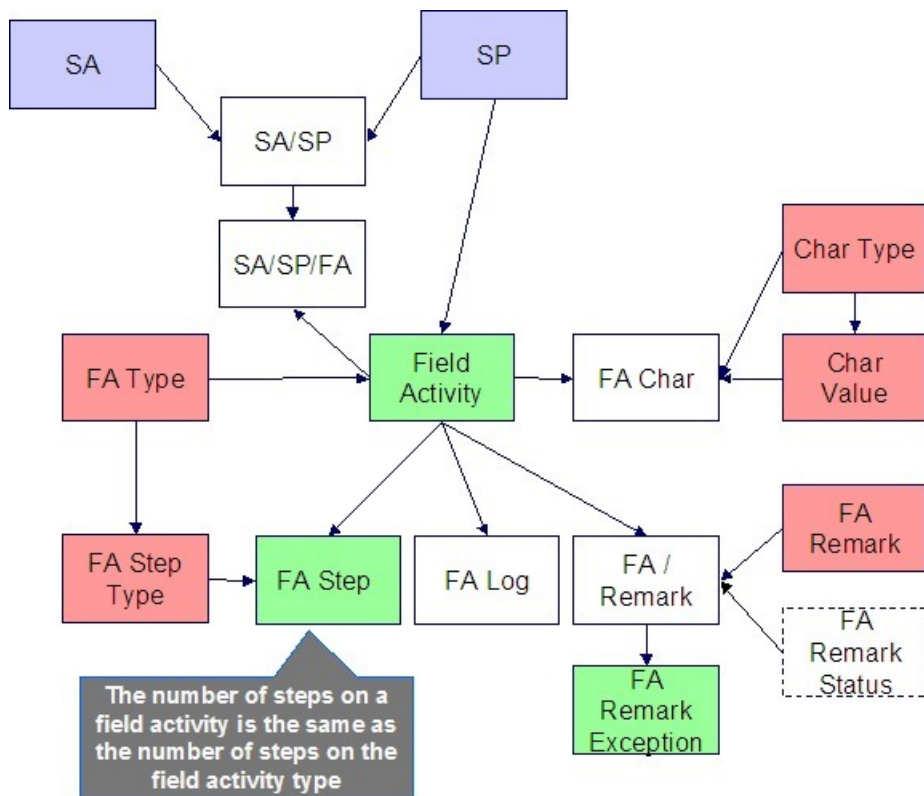
Field Order Suggestions

N/A

Field Activity

Field Activity Data Model

The following data model illustrates the field activity object.



Field Activity Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Field Order	CI_FO	Yes CI_FO_K	VAL-FO	CIPVFORV	CIPVFORK Has dependencies	CIPVFORI
Field Activity	CI_FA	Yes CI_FA_K	VAL-FA		CIPVFACK Has dependencies	CIPVFACI
Step	CI_FA_STEP	No. The key is FA_ID plus sequence number.		CIPVFSTV		CIPVFSTI
Characteristic	CI_FA_CHAR	No. The key is FA_ID		CIPVFAHV		CIPVFAHI

plus CHAR_
TYPE_CD
plus sequence
number.

Remarks	CI_FA_REM	No. The key is FA_ID plus FA_REM_CD.	CIPVFARV	CIPVFARI
Log	CI_FA_LOG	No. The key is FA_ID plus sequence number.	CIPVFALV	CIPVFALI
SA/SP/FA	CI_SA_SP_FA	No. The key is SA/SP id and FA id.	CIPVSSFV	CIPVSSFI

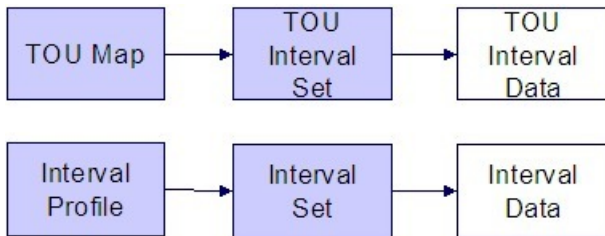
Field Activity Suggestions

N/A

Interval Data

Interval Data Data Model

The following data model illustrates the SA Interval Billing object.



Interval Data Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
TOU Interval Set	CI_TOU_DATA_SET	Yes CI_TOU_DATA_SET_K	VAL-TDS		CIPVTDSK Has dependencies	CIPVTDSI

TOU Interval Data	CI_TOU_DATA	No. The key is TOU_DATA_SET_ID plus TOU_DATA_DTTM		CIPVTOFV	CIPVTODI
Interval Set	CI_INTV_DATA_SET	Yes CI_INTV_DATA_SET_K	VAL-IDS	CIPVIDSK Has dependencies	CIPVIDSI
Interval Data	CI_INTV_DATA	No. The key is INTV_DATA_SET_ID plus INTV_DATA_DTTM		CIPVITFV	CIPVITDI

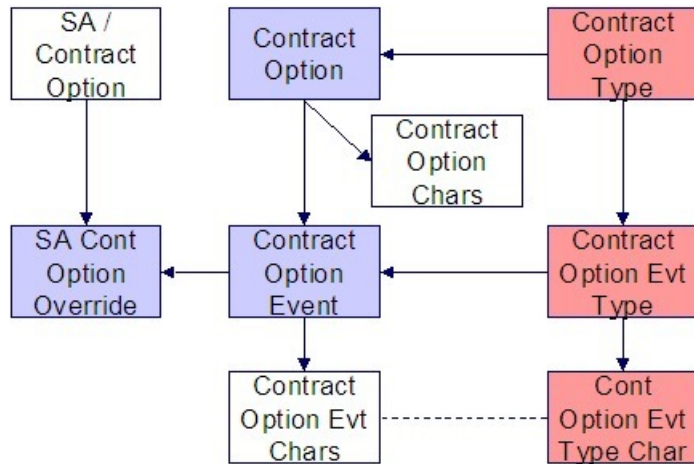
Interval Data Suggestions

N/A

Contract Option Events

Contract Option Events Data Model

The following data model illustrates the Contract Options objects.



Contract Option Events Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity	Key Assignment Batch Control	Insertion Batch Control
-----------------	------------	----------------	---------------------------------	-----------------------	------------------------------	-------------------------

Contract Option Event	CI_COP_EVT	Yes CI_COP_EVT_K	VAL-CEVT	CIPVCEVK Has dependencies	CIPVCEVI
Contract Option Event Characteristics	CI_COP_EVT_CHAR	No. The key is CONT_OPT_EVT_ID plus CHAR_TYPE_CD		CIPVCVCV	CIPVCVCI
SA Contract Option Override	CI_SA_COP_OVRD	No. The key is SA_CONT_OPT_ID plus CONT_OPT_EVT_ID plus OVRD_DTTM		CIPVSCOV	CIPVSCOI

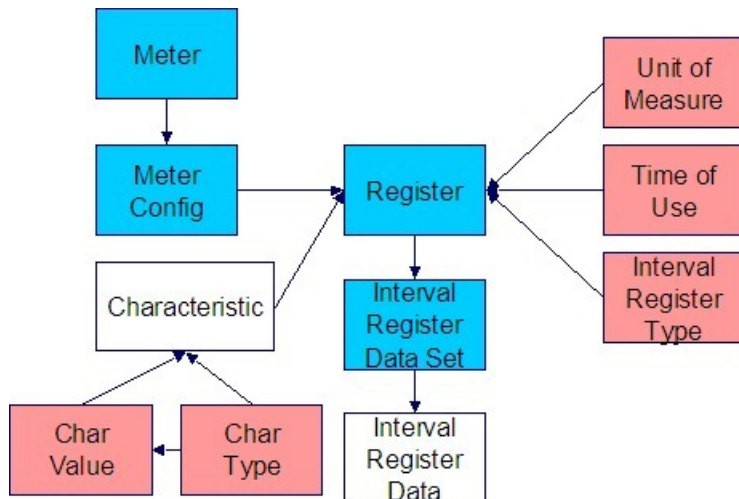
Contract Option Events Suggestions

N/A

Register Interval Data

Register Interval Data Model

The following data model illustrates the register interval data object.



Register Interval Data Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Interval Register Data Set	CI_REG_DATA_SET	Yes CI_REG_DATA_SET_K	VAL-IRDS		CIPVIRSK Has dependencies	CIPVIRSI
Interval Register Data	CI_REG_DATA	No. The key is REG_DATA_SET_ID plus REG_DATA_DTTM		CIPVREFV		CIPVREDI

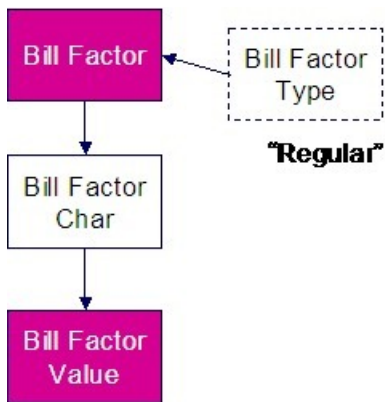
Register Interval Data Suggestions

N/A

Bill Factor Value

Bill Factor Value Data Model

The following data model illustrates the bill factor objects.



Bill Factor Value Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Bill Factor Value	CI_BF_VAL	No. The key is BF_ CD plus CHAR_ TYPE_CD plus CHAR_VAL plus TOU_GRP_CD plus EFFDT	CIPVBFVV		CIPVBFVI (Not threadable)

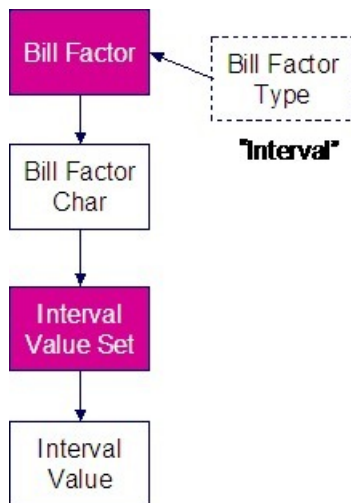
Bill Factor Value Suggestions

N/A

Bill Factor Interval Prices

Bill Factor Interval Prices Data Model

The following data model illustrates the bill factor interval prices objects.



Bill Factor Interval Prices Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Interval Value Set	CI_INTV_VAL_ SET	Yes CI_INTV_ VAL_SET_K	VAL-IVS		CIPVIVSK	CIPVIVSI

Interval Value

CI_INTV_VAL

No. The key is
INTV_VAL_SET_
ID plus INTV_
VAL_DTTM

CIPVITFV

CIPVITVI

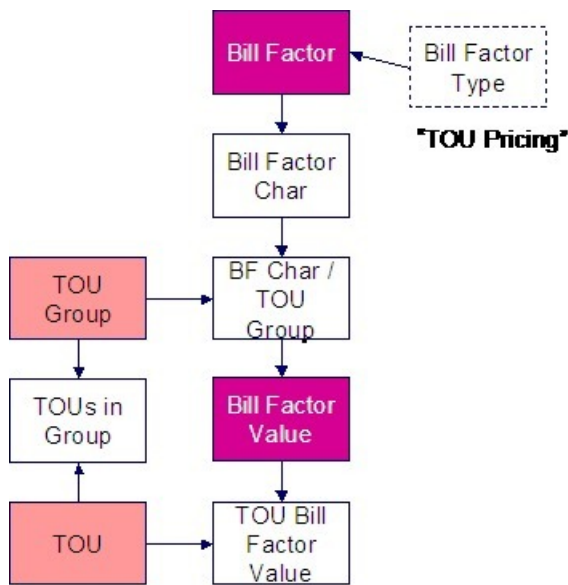
Bill Factor Interval Prices Suggestions

N/A

Bill Factor TOU Pricing

Bill Factor TOU Pricing Data Model

The following data model illustrates the bill factor TOU Pricing objects.



Bill Factor TOU Pricing Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
TOU Bill Factor Value	CI_TOU_BF_VAL	No. The key is BF_ CD plus CHAR_ TYPE_CD plus CHAR_VAL plus EFFDT plus TOU_	CIPVTBVV		CIPVTBVI (Not threadable)

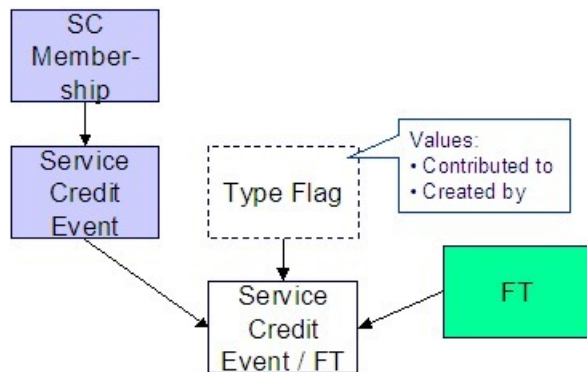
Bill Factor TOU Pricing Suggestions

N/A

Service Credit Event

Service Credit Event Data Model

The following data model illustrates the Service Credit Event objects.



Service Credit Event Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Service Credit Event	CI_SC_EVT	Yes CI_SC_EVT_K	CIPVSCVV	CIPVSCVK Has dependencies	CIPVSCVI
Service Credit Event / FT	CI_SC_EVT_FT	Yes CI_SC_EVT_ FT_K	CIPVSCFV	CIPVSCFK Has dependencies	CIPVSCFI

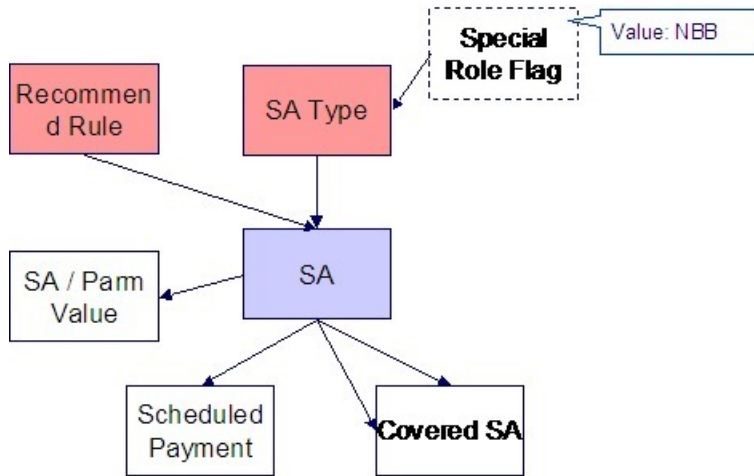
Service Credit Event Suggestions

Loading the Service Credit Event FT data is optional. This data need only be converted if it exists in the legacy system and it is deemed necessary to include it.

Non-Billed Budgets

Non-Billed Budgets Data Model

The following data model illustrates the Non-Billed Budgets objects.



Non-Billed Budgets Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
NBB / Service Agreement	CI_NB_SA	No. The key is SA_ID plus CVRD_SA_ID	CIPVNBSV		CIPVNBSI
NBB Scheduled Payments	CI_NB_SCHED_PAY	Yes CI_NB_SCHED_PAY_K	CIPVNSPV	CIPVNSPK Has dependencies	CIPVNSPI
NBB / SA Parameters	CI_SA_NB_PARM	No. The key is SA_ID plus Sequence Number.	CIPVNPMV		CIPVNPMI

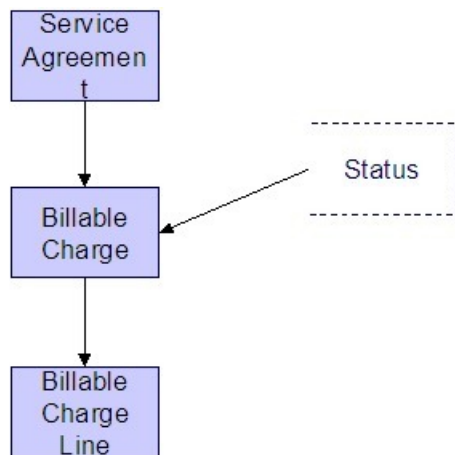
Non-Billed Budgets Suggestions

N/A

Billable Charge

Billable Charge Data Model

The following data model illustrates the Billable Charge objects.



Billable Charge Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Billable Charge	CI_BILL_CHG	Yes. CI_BILL_CHG_K	VAL-BCHG	CIPVBCGV	CIPVBCGK	CIPVBCGI
Billable Charge Line	CI_B_CHG_LINE	No. The key is billable charge id and a sequence number		CIPVBCLV		CIPVBCLI

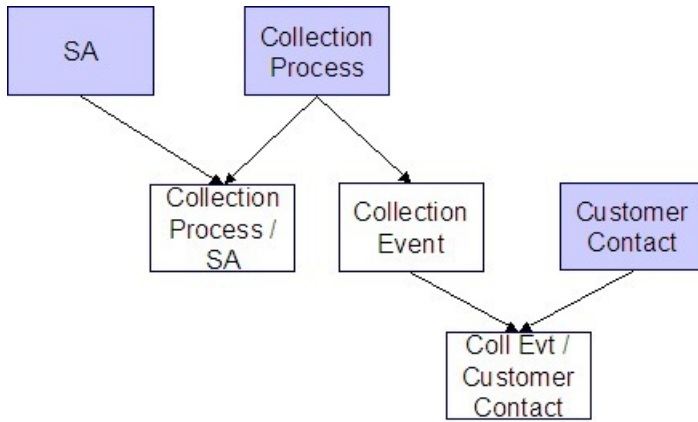
Billable Charge Suggestions

N/A

Collection Process

Collection Process Data Model

The following data model illustrates the Collection Process objects.



Collection Process Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Collection Process	CI_COLL_PROC	Yes. CI_COLL_ PROC_K	VAL-COLL	CIPVCLPV	CIPVCLPK	CIPVCLPI
Collection Event	CI_COLL_EVT	No. The key is collection process id and a sequence number.		CIPVCVNV		CIPVCVNI
Collection Process / Service Agreement	CI_COLL_ PROC_SA	No. The key is collection process id and service agreement id.		CIPVCLSV		CIPVCLSI
Collection Event Customer Contact	CI_COLL_EVT_ CC	No. The key is collection process id, a sequence number and the customer contact id.		CIPVCECV		CIPVCECI

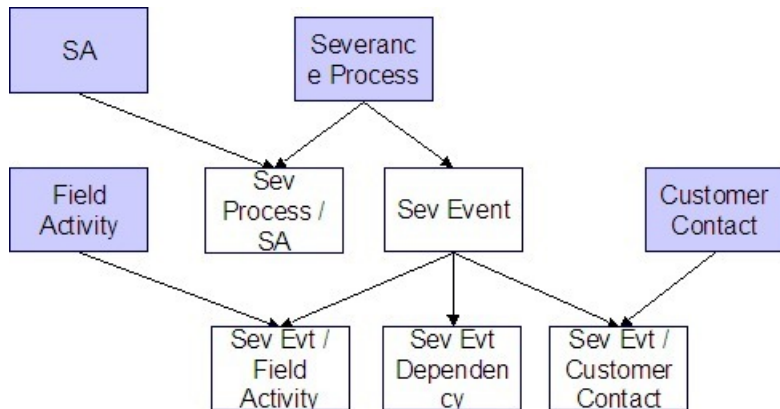
Collection Process Suggestions

N/A

Severance Process

Severance Process Data Model

The following data model illustrates the Severance Process objects.



Severance Process Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Severance Process	CI_SEV_PROC	Yes. CI_SEV_PROC_K	VAL-SEVP	CIPVSEPV	CIPVSEPK	CIPVSEPI
Severance Event	CI_SEV_EVT	No. The key is severance process id and a sequence number.		CIPVSEVV		CIPVSEVI
Severance Event Customer Contact	CI_SEV_EVT_CC	No. The key is severance process id, a sequence number and the customer contact id.		CIPVSECV		CIPVSECI
Severance Event / Field Activity	CI_SEV_EVT_FA	No. The key is severance process id, a sequence number and field activity id.		CIPVSEFV		CIPVSEFI
Severance Event Dependency	CI_SEV_EVT_DEP	No. The key is severance process id, event sequence		CIPVSEDV		CIPVSEDI

number and
a sequence
number.

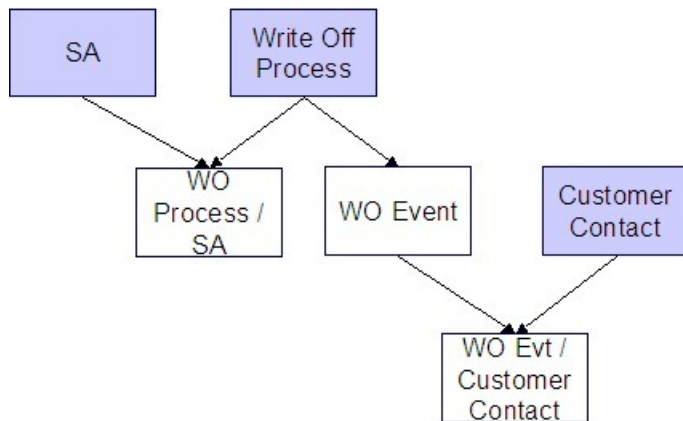
Severance Process Suggestions

N/A

Write Off Process

Write Off Process Data Model

The following data model illustrates the Write Off Process objects.



Write Off Process Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Write Off Process	CI_WO_PROC	Yes. CI_WO_PROC_K	VAL-WOP	CIPVWOPV	CIPVWOPK	CIPVWOPI
Write Off Process / Service Agreement	CI_WO_PROC_SA	No. The key is write-off process id and service agreement id.		CIPVWOSV		CIPVWOSI
Write Off Event	CI_WO_EVT	No. The key is write-off process id and		CIPVWOV		CIPVWOVI

		a sequence number.		
Write Off Event / Customer Contact	CI_WO_EVT_CC	No. The key is write-off process id, a sequence number and the customer contact id.	CIPVWOCV	CIPVWOCI

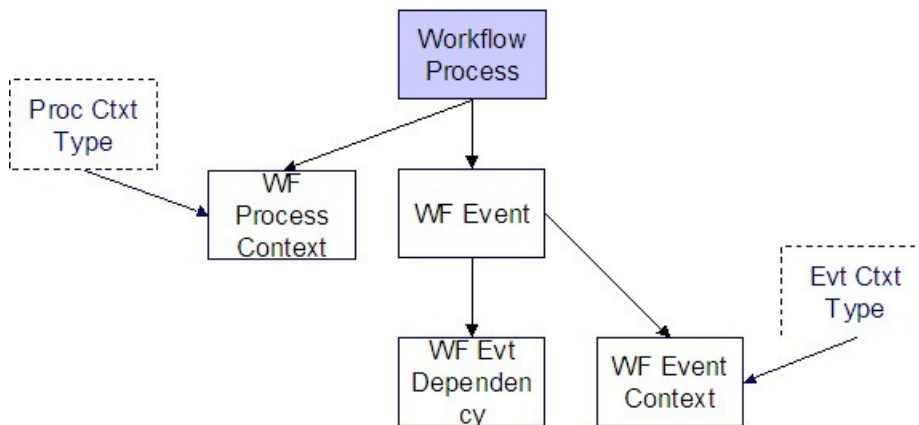
Write Off Process Suggestions

N/A

Workflow Process

Workflow Process Data Model

The following data model illustrates the Workflow Process objects.



Workflow Process Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Workflow Process	CI_WF_PROC	Yes. CI_WF_PROC_K	VAL-WFP	CIPVWPRV	CIPVWPRK	CIPVWPRI
Workflow Process Context	CI_WF_PROC_CTXT	No. The key is workflow process id, workflow		CIPVWPCV		CIPVWPCI

		process context type and value.		
Workflow Event	CI_WF_EVT	No. The key is workflow process id and a sequence number.	CIPVWEV	CIPVWEVI
Workflow Event Context	CI_WF_EVT_CTX	No. The key is workflow process id, event sequence number, event context type and value.	CIPWECV	CIPWECI
Workflow Event Dependency	CI_WF_EVT_DEP	No. The key is workflow process id, event sequence number and a sequence number.	CIPWEDV	CIPWEDI

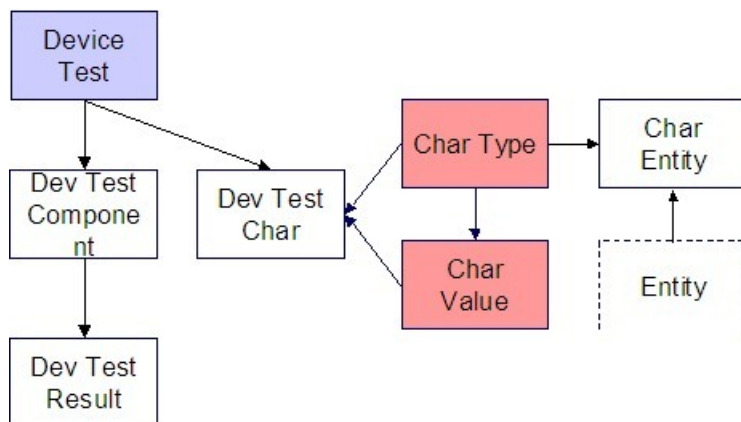
Workflow Process Suggestions

N/A

Device Test

Device Test Data Model

The following data model illustrates the Device Test object.



Device Test Table Names

Data Model Name	Table Name	Generated Keys	Object Validation Batch Control	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Device Test	CI_DV_TEST	Yes. CI_DV_TEST_K	VAL-DTST	CIPVDTTV	CIPVDTTK	CIPVDTTI
Device Test Characteristics	CI_DV_TEST_CHAR	No. The key is device test id and characteristic type code		CIPVDTCV		CIPVDTCI
Device Test Component	CI_DV_TEST_COMP	No. The key is device test id and a sequence number.		CIPVDTMV		CIPVDTMI
Device Test Result	CI_DV_TEST_RES	No. The key is device test id, sequence number, test component type code and a sequence number		CIPVDTRV		CIPVDTRI

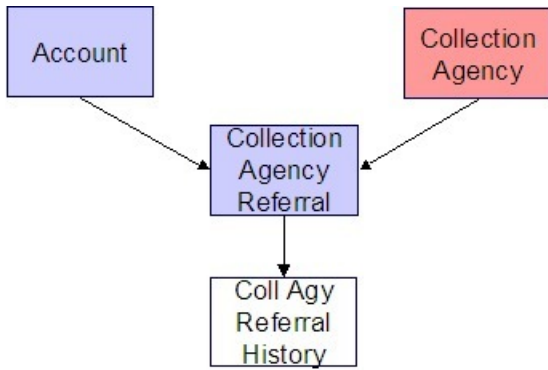
Device Test Suggestions

N/A

Collection Agency Referral

Collection Agency Referral Data Model

The following data model illustrates the Collection Agency Referral object.



Collection Agency Referral Table Names

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Collection Agency Referral	CI_COLL_AGY_REF	Yes. CI_COLL_AGY_REF_K	CIPVCARV	CIPVCARK	CIPVCARI
Collection Agency Referral History	CI_COLL_AGY_HIS	No. The key is collection agency referral id and characteristic type code	CIPVARHV		CIPVARHI

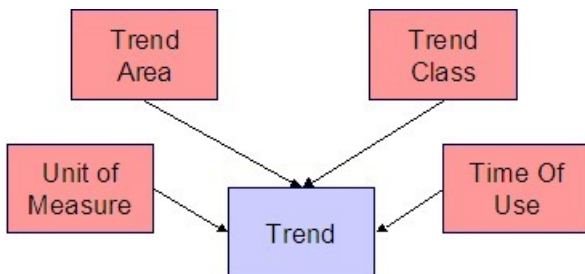
Collection Agency Referral Suggestions

N/A

Trend

Trend Data Model

The following data model illustrates the Trend object.



Trend Table Name

Data Model Name	Table Name	Generated Keys	Referential Integrity Validation Batch Control	Key Assignment Batch Control	Insertion Batch Control
Trend	CI_TREND	No. The key is batch run number, trend area code, trend class code, unit of measure code, time of use code and read date.	CIPVTRNV		CIPVTRNI

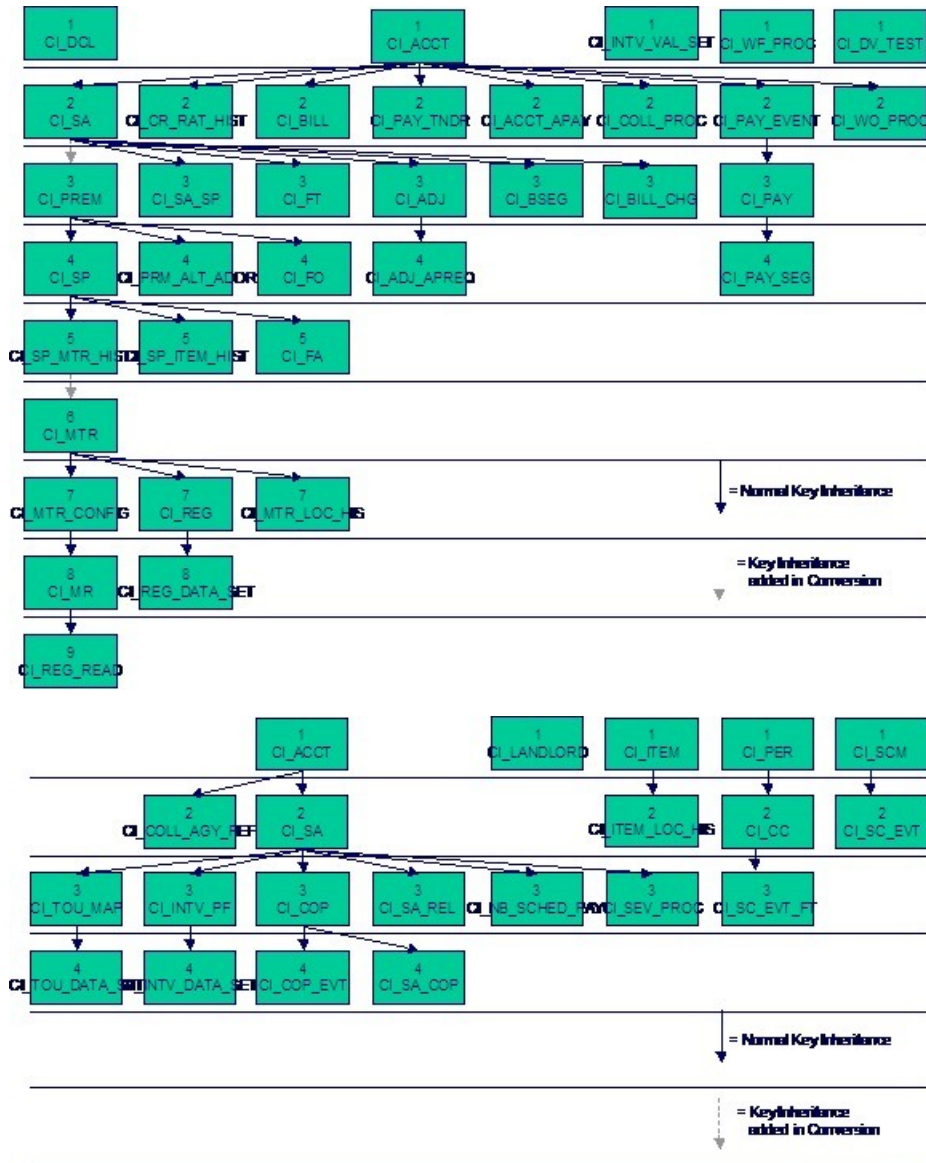
Trend Suggestions

N/A

Program Dependencies

The programs used to assign production keys are listed under [Master Data](#) and [Transaction Data](#) (in the Table Names matrices). Most of these programs have no dependencies (i.e., they can be executed in any order you please). The only exceptions to this statement are illustrated in the following diagram.

The tiers in this diagram contain a box for each table whose key assignment program is dependent on the successful execution of other key assignment programs. The numbers that appear in the boxes describe the order in which these programs must be executed.



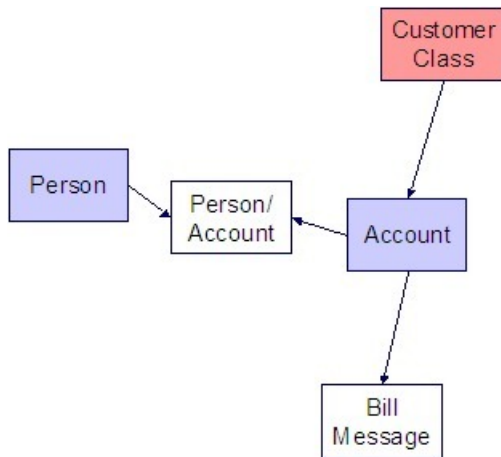
Please refer to the various "Table Names" sections above for the respective names of the programs to allocate each table's keys.

WARNING:

Prior to running the key generation program for a particular object, it is required that any previously generated keys be cleared from the key allocation tables and the key allocation temporary storage table. It is recommended that the key allocation tables be analyzed between runs to maximize performance.

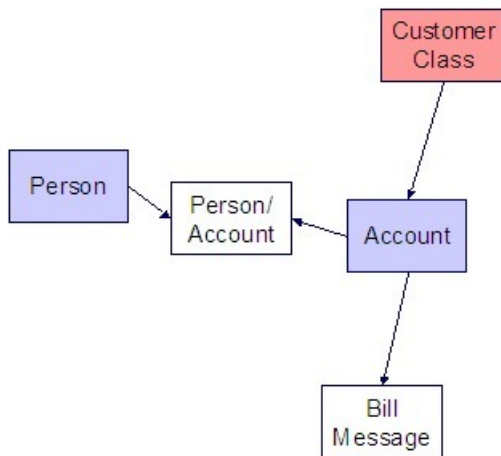
Appendix A - Entity Relationship Diagramming Standards

Because all data is stored in relational table, you need to be able to read diagrams that illustrate relationships between the various tables. The following entity diagram uses every diagramming notation used in the documentation:



Relationships

The solid line connecting the two entities that is terminated by an arrow represents a relationship between two entities. You read the relationship from the entity without the arrow to the entity with the arrow. For example in the following diagram, the line between Customer Class and Account illustrates that a Customer Class may have many Accounts, but an Account may be part of a single customer class.



Entity

Every box on the above diagram represents an entity (i.e., a table). An entity may be a physical entity, such as a Person, or a logical construct, such as an Account.

Color Coding













If you can view this document in color, you will notice that each entity is colored. The color indicates the "subsystem" which governs the entity. Know the governing subsystem is important because:

- The system's menu structure is subsystem-oriented (i.e., if you know the subsystem, you will know how to use the menus to navigate to the page used to view and update the entity).

- The system's documentation is subsystem-oriented (i.e., if you know the subsystem, you will know which chapter contains information about the entity).

Some entities are not color-coded (i.e., they are white). These entities do not have a dedicated page, as they are part of a parent entity. For example, the Person / Account entity above is related to the Account object and does not have its own page. You must display the parent entity in order to view such an entity. For example, if you want to look at Person / Account information, you must go to the Account page.

The following table describes the colors utilized in the documentation:

Color	Subsystem
	Customer Information
	Admin (Control) Table. These tables are referenced as foreign keys on master and transaction tables. We do not document the names of these tables in this document as the table names are easily accessible using the Table transaction.
	N/A - the entity is maintained in respect of a higher level entity.
	N/A - the values in these types of entities are defined in a special table referred to as the lookup table. In order to determine the valid values for a column that references a lookup table, use the name of the column as the search value on the Look Up user interface.
	Meter Management
	Meter Reading
	Rates
	Billing
	Financial Transaction
	Payment
	Field Order
	Adjustment

Appendix B - Multiple Owners In A Single Database

In the schematic referenced in the [Introduction](#), you'll notice that there are two table owners in the system database. We refer to the first owner as "staging" and the second owner as "production".

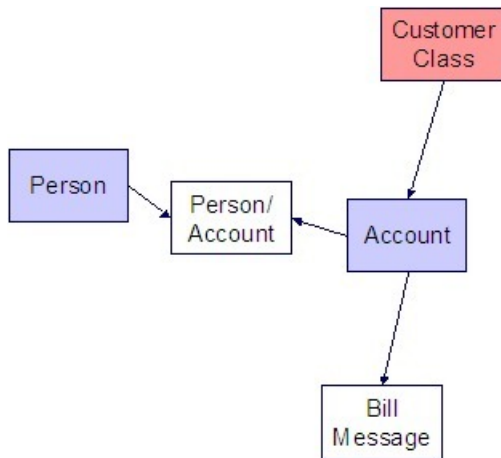
The staging owner is linked to the tables into which you insert your pre-validated data. These tables have an owner ID of CISSTG.

NOTE:

Multiple staging databases. It is possible to have multiple staging databases. In this situation, each one would have a unique owner ID, e.g., CISSTG1, CISSTG2, etc.

The production owner is linked to the tables used by your production system. These tables have an owner ID of CISADM.

When the validation programs run against your staging data, they validate the staging data against the production control tables (and insert errors into the staging error table). This means that the SQL statements that access / update master and transaction data need to use the staging owner (CISSTG). Whereas the SQL statements that access control tables need to use the production owner (CISADM).



But notice that when these same programs run against production (Validate (b)), the SQL statements will never access the staging owner. Rather, they all point at the production owner.

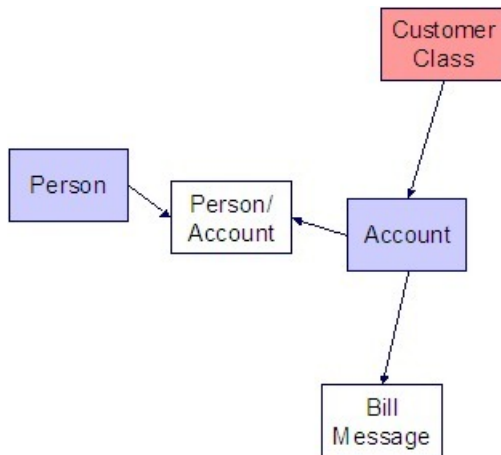
This is accomplished as follows:

- A separate application server must exist for each owner. Each application server points at a specific database user ID.
- The database user ID associated with the staging database uses CISSTG as the owner for the master and transaction tables, but it uses CISADM as the owner of the production control tables.
- The database user ID associated with the production database uses CISADM as the owner for the master, transaction, and control tables.

You may wonder why we went to this trouble. There are several reasons:

- We wanted to reuse the validation logic that exists in the programs that validate your production data. In order to achieve this, these programs must sometimes point at the staging owner, and other times they must point at the production owner (and this must be transparent to the programs otherwise two sets of SQL would be necessary).
- We wanted to let you use the application to look at and correct staging data. This can be accomplished by creating an application server that points at your staging database with the ownership characteristics described above.
- We wanted the validation programs to be able to validate your production data (in addition to your staging data). Why would you want to validate production data if only clean data can be added to production? Consider the following scenarios:
 - After an upgrade, you might want to validate your production data to ensure your pre-existing user-exit logic still works.
 - You may want to conduct an experiment of the ramifications of changing your validation logic. To do this, you could make a temporary change to user exit logic (in production) and then run the validation jobs on a random sample basis.
 - You forget to run a validation program before populating production and you want to see the damage. If you follow the instructions in this document, this should never happen. However, accidents happen. And if they do, at least there's a way to determine the ramifications.

While the redirection of owner ID's is a useful technique for the validation programs, it cannot be used by the key assignment and production insert programs? Why, because these programs have to access the same tables but with different owners. For example, the program that inserts rows into the person table must select rows from staging.Person and insert them into production.Person.



This is accomplished as follows:

- Views exist for each table that exists in both databases. These views have hard-coded the database owner CISADM (production). For example, there is a view called CX_PER that points at person table in production.
- The key assignment and insertion programs use these views whenever then need to access production data.

Appendix C - Known Oddities

Be aware that the following tables reference master data (e.g., persons, accounts). This means that if you look at them using a user ID that defaults ownership to the staging level, you will not be able to see the related master data (because the person / account doesn't exist in the staging owner's tables).

- Collection Agency. References a person.
- Service Provider. References a person and a service agreement.
- 3 rd Party Payor. References an account.
- Tender Source. References a suspense account.