Oracle Utilities Meter Solution

Administrative User Guide

Release 2.3.0.2.2

F21772-05

August 2020

ORACLE

Oracle Utilities Meter Solution Administrative User Guide

Release 2.3.0.2.2

F21772-05

August 2020

Documentation build: 6.18.2020 16:15:32 [D1_1592514932000]

Copyright © 2010, 2020, 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.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

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

Framework Administrative User Guide....22
General Options.... 22
Defining Installation Options.... 22
Installation Options - Main....22
Installation Options - Messages.... 23
Installation Options - Algorithms.... 23
Installation Options - Accessible Modules.... 25
Installation Options - Installed Products....25
Support For Different Languages.... 26
User Language.... 26
Customer Language.... 26
Defining Languages....27
Defining Countries....28
Country - Main....28
Country - States.... 28
Defining Currency Codes....28
Defining Time Zones....29
Designing Time Zones.... 29
Setting Up Time Zones.... 31
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
Defining Feature Configurations.... 35
Feature Configuration - Main.... 35
Feature Configuration - Messages....36
Defining Master Configurations....36
Security & User Options.... 37
The Big Picture of Application Security....37
Application Security.... 37
Action Level Security....39
Field Level Security....39
Encryption and Masking....40
System Encryption.... 40
User Interface Masking.... 40
Application Encryption....45
The Base Package Controls One User, One User Group, And Many Application Services....47
Importing Security Configuration from an External Source....47
The Big Picture of Row Security....48
Defining Application Services.... 48
Defining Security Types....49
Security Type - Main....49
Defining User Groups.... 49
User Group - Main.... 50
User Group - Application Services....50
User Group - Users....51
User Group Services Management.... 51
Defining Access Groups.... 52
Defining Data Access Roles....52
Data Access Role - Main.... 52
Data Access Role - Access Group.... 52
Defining Users....53
Data Privacy....53
The Approach to Implementing Object Erasure....53
Configuring a Maintenance Object for Erasure.... 54
Manage Erasure Schedule Algorithm.... 54

Monitoring the Schedule and Performing Erasure....55
Erasing User Information By Obfuscation....55
Viewing an Object's Erasure Status....55
Viewing Erasure Configuration....55
Archiving the Object Erasure Schedule....56
Cryptography Keys....56
Understanding Key Rings....56
Signature Keys....56
Defining Key Rings....57
User Interface Tools....57
Defining Menu Options....57
Menu - Main....57
Menu - Menu Items....58
The Big Picture of System Messages....60
Defining System Messages....60
Message - Main....60
Message - Details....61
The Big Picture of Portals and Zones....62
There Are Three Types of Portals....62
Common Characteristics of All Portals....62
Portals Are Made Up of Zones....62
Configuring Zones for a Portal....62
Granting Access to Zones....63
Common Characteristics of Stand-Alone Portals....64
Putting Portals on Menus....64
Granting Access to A Portal....64
Custom Look and Feel Options....65
User Interface....65
UI Map Help....65
Setting Up Portals and Zones....65
Defining Zone Types....66
Defining Zones....67
Zone - Main....67
Zone - Portal....68
Zone Configuration Topics....68
Defining Context-Sensitive Zones....92
Defining Portals....92
Portal - Main....92
Portal - Options....94
Defining Display Icons....94
Defining Navigation Keys....94
Navigation Key Types....94
Navigation Key vs. Navigation Option....95
The Flexibility of Navigation Keys....95
Linking to External Locations....95
Overriding Navigation Keys....96
Maintaining Navigation Keys....96
Defining Navigation Options....97
Navigation Option - Main....98
Navigation Option - Tree....99
Database Options....100
Understanding The Data Model....100
Fields....101
Tables....102
Maintenance Objects....104
Defining Valid Values....105
Characteristic Types....106
Foreign Key References....108
Defining Tables....110
Table - Main....110
Table - Table Field....112
Table - Constraints....113
Table - Referred by Constraints....114
Defining Fields....114

Defining Maintenance Objects....114
Maintenance Object - Main....115
Maintenance Object - Options....115
Maintenance Object - Algorithms....115
Maintenance Object - Maintenance Object Tree....117
Defining Lookups....117
Lookup - Main....118
Defining Extendable Lookups....119
Extendable Lookup Advanced Topics....119
Defining Characteristic Types....121
Characteristic Type - Main....121
Characteristic Type - Characteristic Entities....122
Defining Foreign Key References....122
Audit Trails....123
Captured Information....123
How Auditing Works....124
The Audit Trail File....124
How To Enable Auditing....124
Turn On Auditing For a Table....125
Specify The Fields and Actions To Be Audited....125
Audit Queries....125
Audit Query by User....126
Audit Query by Table / Field / Key....126
Bundling....127
About Bundling....127
Sequencing of Objects in a Bundle....128
Recursive Key References....128
Owner Flags on Bundled Entities....128
Configuring Maintenance Objects for Bundling....128
Making Maintenance Objects Eligible for Bundling....129
Adding a Foreign Key Reference....129
Creating a Physical Business Object....129
Creating a Bundling Add Business Object....130
Adding the Current Bundle Zone....130
Adding a Customized Entity Search Query Zone to the Bundle Export Portal....130
Working with Bundles....131
Creating Export Bundles....131
Creating and Applying Import Bundles....132
Editing Export Bundles....132
Editing Import Bundles....133
Revision Control....133
About Revision Control....133
Turning On Revision Control....133
Configuring Maintenance Objects for Revision Control....134
Working with the Revision Control Zones....135
Checking Out an Object....135
Checking In an Object....135
Reverting Changes....136
Forcing a Check In or Restore....136
Deleting an Object....136
Restoring an Object....136
Working with the Revision Control Portal....137
Revision Control Search....137
Information Lifecycle Management 138
The Approach to Implementing Information Lifecycle Management....138
Batch Processes....138
Eligibility Algorithm....140
Enabling ILM for Supported Maintenance Objects....140
Ongoing ILM Tasks....141
Archived Foreign Keys....141
Configuration Tools....142
Business Objects....142
The Big Picture of Business Objects....142
What Is A Business Object?....142

Business Object Inheritance ... 148
Each Business Object Can Have A Different Lifecycle ... 149
BO Algorithm Execution Summary ... 156
Granting Access To Business Objects ... 157
Defining Business Objects ... 157
Business Object - Main ... 157
Business Object - Schema ... 158
Business Object - Algorithms ... 159
Business Object - Lifecycle ... 161
Business Object - Summary ... 163
Advanced BO Tips and Techniques ... 163
Managing To Do Entries ... 163
Submitting a Batch Job ... 164
Defining Status Reasons ... 164
Data Areas ... 165
Defining Data Areas ... 165
Data Area - Main ... 165
Data Area - Schema ... 166
Business Services ... 166
Service Program ... 167
Defining Business Services ... 167
Business Service - Main ... 167
Business Service - Schema ... 168
Useful Services and Business Services ... 168
User Interface (UI) Maps ... 172
Defining UI Maps ... 174
UI Map - Main ... 174
UI Map - Schema ... 175
UI Map Attributes and Functions ... 175
UI Map Standards ... 234
Ensuring Unique Element IDs for UI Maps ... 245
Process Flows ... 246
Understanding Process Flows ... 246
Designing Process Flows ... 248
Defining Process Flow Types ... 250
Algorithms ... 250
The Big Picture Of Algorithms ... 250
How To Add A New Algorithm ... 252
Minimizing The Impact Of Future Upgrades ... 252
Defining Algorithm Types ... 253
Defining Algorithms ... 254
Script Options ... 255
The Big Picture Of Scripts ... 255
Scripts Are Business Process-Oriented ... 255
A Script Is Composed Of Steps ... 255
A Script May Declare Data Areas ... 255
Securing Script Execution ... 256
The Big Picture Of BPA Scripts ... 256
How To Invoke Scripts ... 257
Developing and Debugging Your BPA Scripts ... 257
Launching A Script From A Menu ... 257
Launching A Script When Starting The System ... 258
Executing A Script When A To Do Entry Is Selected ... 258
The Big Picture Of Script Eligibility Rules ... 258
The Big Picture Of Server-Based Scripts ... 262
Additional Coding Options For Server Scripts ... 263
Plug-In Scripts ... 264
Service Scripts ... 265
Groovy Library Scripts ... 266
Debugging Server-Based Scripts ... 266
Maintaining Scripts ... 267
Script - Main ... 267
Script - Step ... 268
Script - Data Area ... 309

Script - Schema ... 309
Script - Eligibility ... 310
Merging Scripts ... 311
Script Merge ... 312
Maintaining Functions ... 314
Function - Main ... 315
Function - Send Fields ... 315
Function - Receive Fields ... 316
Maintaining Managed Content ... 316
Managed Content - Main ... 316
Managed Content - Schema ... 317
Advanced Schema Topics ... 317
Schema Nodes and Attributes ... 317
UI Hint Syntax ... 337
Schema Designer ... 345
Schema Viewer ... 347
Business Event Log ... 347
Miscellaneous Topics ... 348
Module Configuration ... 348
Menu Item Suppression ... 348
Menu Suppression ... 348
Turn Off A Function Module ... 349
Global Context Overview ... 349
System Data Naming Convention ... 349
Base Product System Data ... 349
Implementation System Data ... 350
Accessibility Considerations ... 350
Referencing URIs ... 350
Validation Against a Whitelist ... 351
URI Substitution ... 351
External File Storage ... 352
Caching Overview ... 353
Expression Parser ... 354
Debug Mode ... 357
System Override Date ... 357
Advanced Search Options ... 358
To Do Lists ... 358
The Big Picture of To Do Lists ... 358
To Do Entries Reference A To Do Type ... 358
To Do Entries Reference A Role ... 359
To Do Entries Can Be Rerouted (Or Suppressed) Based On Message Number ... 359
The Priority Of A To Do Entry ... 360
Working On A To Do Entry ... 361
Monitoring A To Do Entry ... 361
Launching Scripts When A To Do Is Selected ... 361
To Do Entries Have Logs ... 361
How Are To Do Entries Created? ... 362
To Do Entries Created By Background Processes ... 362
To Do Entries Created By Algorithms ... 363
To Do Entries Created Manually ... 364
The Lifecycle Of A To Do Entry ... 365
Linking Additional Information To A To Do Entry ... 365
Implementing Additional To Do Entry Business Rules ... 366
To Do Entries May Be Routed Out Of The System ... 366
To Do Information May Be Formatted By An Algorithm ... 366
Periodically Purging To Do Entries ... 367
Setting Up To Do Options ... 367
Installation Options ... 367
Messages ... 367
Feature Configuration ... 368
Defining To Do Roles ... 368
To Do Role - Main ... 368
To Do Role - To Do Types ... 369
Defining To Do Types ... 369

To Do Type - Main ... 369
To Do Type - Roles ... 370
To Do Type - Sort Keys ... 370
To Do Type - Drill Keys ... 371
To Do Type - Message Overrides ... 371
To Do Type - To Do Characteristics ... 371
To Do Type - Algorithms ... 372
Implementing The To Do Entries ... 373
Background Processes ... 373
Understanding Background Processes ... 373
Background Processing Overview ... 374
Parallel Background Processes ... 375
Optimal Thread Count ... 375
Parameters Supplied To Background Processes ... 376
Indicating a File Path or File Name ... 378
Manifest File ... 379
Processing Errors ... 380
Error Post-Processing Logic ... 380
Post-Processing Logic ... 381
Timed Batch Processes ... 381
Monitor Background Processes ... 381
Plug-in Driven Background Processes ... 383
Processing System Records ... 383
Uploading Records ... 388
How to Re-extract Information ... 389
How to Submit Batch Jobs ... 390
How to Track Batch Jobs ... 390
How to Restart Failed Jobs and Processes ... 390
Assessing Level of Service ... 390
Defining Batch Controls ... 390
Batch Control - Algorithms ... 393
Defining File Integration Types ... 394
On-Line Batch Submission ... 394
Batch Submission Creates a Batch Run ... 394
Jobs Submitted in the Background ... 395
Email Notification ... 395
Running Multi-Threaded Processes ... 395
Batch Jobs May End in Error ... 395
Submitting Jobs in the Future ... 396
Lifecycle of a Batch Job Submission ... 396
Granting Access To Batch Submission ... 396
Batch Job Submission - Main ... 396
Tracking Batch Processes ... 398
Batch Run Tree - Main ... 398
Batch Run Tree - Run Control ... 399
The Big Picture of Requests ... 400
Request Type Defines Parameters ... 400
Previewing and Submitting a Request ... 400
To Do Summary Email ... 400
Defining a New Request ... 401
Setting Up Request Types ... 401
Maintaining Requests ... 401
Attachments ... 402
Attachment Overview ... 402
Configuring Your System for Attachments ... 403
Maintaining Attachments ... 404
Adding Attachments ... 404
Application Viewer ... 405
Application Viewer Toolbar ... 405
Data Dictionary Button ... 405
Physical and Logical Buttons ... 405
Collapse Button ... 405
Attributes and Schema Button ... 406
Maintenance Object Button ... 406

Algorithm Button ... 406
Batch Control Button ... 406
To Do Type Button ... 406
Description and Code Buttons ... 407
Service XML Button ... 407
Select Service Button ... 407
Java Docs Button ... 407
Groovy Java Docs Button ... 408
Classic Button ... 408
Preferences Button ... 408
Help Button ... 408
About Button ... 408
Slider Icon ... 408
Data Dictionary ... 409
Using the Data Dictionary List Panel ... 409
Primary And Foreign Keys ... 409
Field Descriptions Shown ... 410
Using the Data Dictionary Detail Panel ... 410
Related Tables View ... 410
Table Detail View ... 410
Column Detail View ... 411
Maintenance Object Viewer ... 411
Using the Maintenance Object List Panel ... 411
Using the Maintenance Object Detail Panel ... 411
Algorithm Viewer ... 412
Using the Algorithm Viewer List Panel ... 412
Using the Algorithm Plug-In Spot Detail Panel ... 412
Using the Algorithm Type Detail Panel ... 412
Using the Algorithm Detail Panel ... 412
Batch Control Viewer ... 412
Using the Batch Control Viewer List Panel ... 413
Using the Batch Control Detail Panel ... 413
To Do Type Viewer ... 413
Using the To Do Type Viewer List Panel ... 413
Using the To Do Type Detail Panel ... 413
Service XML Viewer ... 414
Using the Service XML Viewer Overview Panel ... 414
Using the Service XML Viewer Detail Panel ... 414
Java Docs Viewer ... 414
Using the Java Docs Viewer List Panel ... 415
Using the Java Package Detail Panel ... 415
Using the Java Interface / Class Detail Panel ... 415
Groovy Java Docs Viewer ... 415
Application Viewer Preferences ... 415
Application Viewer Stand-Alone Operation ... 416
Stand-Alone Configuration Options ... 416
Example Application Viewer Configuration ... 416
Application Viewer Generation ... 417
Monitoring and Analytics ... 417
Reporting Tool Integration ... 418
The Big Picture Of Reports ... 418
Integration with BI Publisher ... 418
How To Request Reports ... 419
Viewing Reports ... 419
Configuring The System To Enable Reports ... 420
Configuring BI Publisher Reports ... 420
Defining Reporting Options ... 420
Defining Report Definitions ... 421
Sample Reports Supplied with the Product ... 422
How to Use a Sample Report Provided with the System ... 422
How To Define A New Report ... 422
Use a Sample Report as a Starting Point ... 423
Publishing Reports in BI Publisher ... 423
Designing Your Report Definition ... 424

Service Health Check ... 426
Health Check Configuration ... 427
Viewing Health Check Results ... 427
Introducing Health Check Conditions ... 428
Measuring Performance ... 428
Understanding Performance Targets ... 428
Performance Target Objects Overview ... 429
Calculating and Displaying Performance Targets ... 429
Performance Target Metrics and Metric Types ... 429
Performance Target Categories and Types ... 430
Performance Targets Define Specific Metrics ... 430
Objects Linked to a Performance Target ... 430
Creating Performance Target Zones ... 431
Setting Up Performance Target Configuration ... 431
Performance Target Category Lookup ... 431
Defining Performance Target Types ... 431
Maintaining Performance Targets ... 432
Capturing Statistics ... 432
Understanding Statistics ... 432
Configuring Your System for Statistics ... 433
Defining and Monitoring Statistics ... 433
Creating Cube Views ... 433
Understanding Cube Viewer ... 434
Cube Viewer Components ... 434
Cube Configuration Components ... 438
Configuring Cube Types ... 438
Maintaining Cube Types ... 439
Cube Type Advanced Topics ... 439
Product Use Metrics ... 440
External Messages ... 440
Incoming Messages ... 440
Inbound Web Services ... 440
Understanding Inbound Web Services ... 440
Configuring Inbound Web Service Options ... 444
Deploying Inbound SOAP Web Services ... 447
Guaranteed Delivery ... 449
Outgoing Messages ... 449
Outbound Messages ... 450
Polling Outbound Messages Using OSB ... 450
Batch Message Processing ... 451
Real Time Messages ... 451
Designing the System for Outbound Messages ... 452
Configuring the System for Outbound Messages ... 455
Managing Outbound Messages ... 465
Web Service Adapters ... 466
Understanding Web Service Adapters ... 466
Setting Up Web Service Adapters ... 467
Sending Email ... 468
Web Service Category ... 468
Defining Web Service Categories ... 469
JMS Message Browser ... 469
Oracle Integration Cloud Catalog ... 469
Web Service Catalog Configuration ... 470
Web Service Catalog Master Configuration ... 470
Maintaining the Web Service Catalog ... 470
XAI Documentation Note ... 471
Integrations ... 471
Creating Application Links in External Applications ... 471
LDAP Integration ... 472
LDAP Integration Overview ... 473
Configuring LDAP Integration ... 474
LDAP Mapping ... 475
Oracle Identity Manager Integration ... 477
Batch Scheduler Integration ... 479

Data Synchronization....479
Understanding Data Synchronization....479
Maintaining Sync Requests....481
Operational Analytics....481
Understanding Operational Analytics....481
Calendar and Time Dimensions....482
Maintaining Bucket Configurations....482
Defining Analytics Options....483
Defining Analytics Tables....483
Defining Analytics Dimensions....483
Defining Analytics Facts....484
Defining Characteristic Mapping....484
Analytics Integration....485
Understanding Analytics Integration....485
Business Flags....486
Understanding Business Flags....486
Standard Name....486
Business Flag Type Defines Behavior for a Standard Name....486
Business Flag Type Algorithms....487
Objects Linked to a Business Flag....487
Impacted Business Process....487
Dates....487
Creating Business Flags....488
Confidence....489
Setting Up Business Flag Configuration....489
Standard Name Category Characteristic Type....489
Business Flag Standard Name Lookup....490
Business Process Lookup....490
Integration Configuration....490
Defining Business Flag Types....490
Maintaining Business Flags....490
Market Transaction Management....491
Understanding Market Transaction Management....491
Configuring Market Transaction Management....491
Defining Market Configurations....491
Defining Market Message Types....492
Defining Market Process Types....492
Maintaining Market Transactions....492
Maintaining Market Processes....492
Maintaining Market Process Events....493
Maintaining Market Messages....493
Configuration Migration Assistant (CMA)....493
Understanding CMA....494
CMA Overview....494
Types of Migrations....495
Migrating Configuration Data....496
Data with System Generated Primary Keys....497
Importing Data that References a User....498
Additional Considerations....498
CMA Configuration....499
Master Configuration - Migration Assistant....499
Migration Plans....499
Defining a Migration Plan....500
Understanding the BO Filtering Process....501
Migration Plans for Objects with XML-Embedded Links....502
Defining a Migration Request....502
Identifying Tables to Exclude From Migrations....503
Configuring Custom Objects for Migration....504
The CMA Execution Process....505
Exporting a Migration....505
Migration Data Set Export....506
Export Lifecycle....506
Importing and Applying a Migration....507
Import Step....508

Compare Step ... 509
Approval Step ... 512
Apply Step ... 512
Adjusting Data Prior to Comparing ... 517
Import Process Summary ... 517
Cancelling a Data Set ... 522
Additional Note Regarding Imports ... 522
Caching Considerations ... 522
Maintaining Import Data ... 523
Running Batch Jobs ... 524
CMA Reference ... 525
Framework-Provided Migration Configuration ... 525
Facts ... 527
Fact Is A Generic Entity ... 528
Fact's Business Object Controls Everything ... 528
Fact Supports A Log ... 528
Conversion ... 528
Understanding The Conversion Process ... 528
Conversion Entities ... 530
Conversion Steps ... 530
Load Legacy Data Into Staging Tables ... 530
Validate Information In The Staging Tables ... 531
Allocate Production Keys ... 532
XML Resolution ... 534
Insert Production Data ... 535
Validate Production ... 535
A Note About Keys ... 536
Multiple Owners In A Single Database ... 536
Conversion Entity Dashboard ... 537
Data Export ... 538
Understanding Data Export ... 538
Generalized Data Export ... 538
Specialized Data Export ... 542
Defining Data Export Control ... 543
Generalized Export Dashboard ... 543
Meter Solution Products Functional Overview ... 544
Architectural Overview ... 547
Naming Conventions ... 548
High Level Functional Areas ... 548
Configuration Setup Sequence ... 549
Additional Resources ... 553
How to Get Support ... 553
Knowledge Base Articles ... 553
Important Articles ... 554
Support Hot Topic Emails ... 554
Embedded Help ... 554
Leveraging Demonstration Data ... 555
Customer User Groups ... 555
Best Practices ... 556
Performance Recommendations ... 556
Initial Measurement Loading Recommendations ... 556
VEE Recommendations ... 556
Usage Transaction Recommendations ... 557
User Interface Recommendations ... 557
SQL Recommendations ... 558
Java Recommendations ... 558
Referencing Master Data by Identifiers ... 558
Understanding Referencing Master Data by Identifiers ... 558
Recommendations for Creating a Production Environment ... 559
System-Wide Options ... 560
Installation Options - Framework ... 560
Configuring Installation Options - Framework ... 560
Feature Configurations ... 561

Configuring Feature Configurations....561
Time Zones....562
Configuring Time Zones....562
Configuring Multi-time Zone Support....562
Defining General Options....564
Units of Measure....564
Understanding Units of Measure....564
Configuring Units of Measure....564
Service Quantity Identifiers....565
Understanding Service Quantity Identifiers....565
Configuring Service Quantity Identifiers....565
Time of Use....565
Understanding Time of Use....565
Configuring Time of Use....565
Service Types....566
Understanding Service Types....566
Configuring Service Types....566
Divisions....566
Understanding Divisions....566
Configuring Divisions....567
Factors....567
Understanding Factors....567
Configuring Factors....567
Markets....568
Understanding Markets....568
Configuring Markets....568
Market Participants....569
Understanding Market Participants....569
Configuring Market Participants....569
Understanding Processing Methods....569
Processing Timetable Types....571
Understanding Processing Timetable Types....571
Configuring Processing Timetable Types....572
Defining Asset Options....573
Asset Activity Types....573
Understanding Asset Activity Types....573
Defining Asset Activity Types....574
Asset Completion Event Types....574
Understanding Asset Completion Event Types....574
Defining Asset Completion Event Types....574
Asset Inbound Communication Types....575
Understanding Asset Inbound Communication Types....575
Defining Asset Inbound Communication Types....575
Asset Types....576
Understanding Asset Types....576
Defining Asset Types....576
Questions....576
Understanding Questions....577
Defining Questions....577
Service History Types....577
Understanding Service History Types....577
Defining Service History Types....578
Configuration Types....579
Understanding Configuration Types....579
Defining Configuration Types....579
Configurations....580
Understanding Configurations....580
Defining Configurations....580
Configuration Reports....580
Understanding Configuration Reports....580
Defining Configuration Reports....581
Defining Location Options....582
Asset Location Types....582

Understanding Asset Location Types....582
Defining Asset Location Types....583
Organization Types....583
Understanding Organization Types....583
Defining Organization Types....583
Organizations....583
Understanding Organizations....583
Defining Organizations....584
Out of Service Location Types....584
Understanding Out of Service Location Types....584
Defining Out of Service Location Types....584
Out of Service Locations....585
Understanding Out of Service Locations....585
Defining Out of Service Locations....585
Defining Device Options....586
Command Sets....586
Understanding Command Sets....586
Configuring Command Sets....586
Manufacturers....587
Understanding Manufacturers....587
Configuring Manufacturers....587
Head End Systems....587
Understanding Head End Systems....587
Understanding SGG Adapter Configuration....588
Configuring Head End Systems....589
Measuring Component Types....589
Understanding Measuring Component Types....589
Configuring Measuring Component Types....594
Measuring Component Comparison Types....594
Understanding Measuring Component Comparison Types....594
Configuring Measuring Component Comparison Types....595
Device Configuration Types....595
Understanding Device Configuration Types....595
Configuring Device Configuration Types....595
Device Types....596
Understanding Device Types....596
Configuring Device Types....597
Defining Device Installation Options....598
Service Point Types....598
Understanding Service Point Types....598
Configuring Service Point Types....599
Service Point Quantity Types....599
Understanding Service Point Quantity Types....599
Configuring Service Point Quantity Types....599
Measurement Cycles....599
Understanding Measurement Cycles....600
Configuring Measurement Cycles....600
Measurement Cycle and Bill Determinants....600
Measurement Cycle Schedules....601
Understanding Measurement Cycle Schedules....601
Configuring Measurement Cycle Schedules....601
Defining Measurement Options....602
Initial Measurement Data....602
Configuring the Initial Measurement Algorithms....602
Configuring Measurement Logging....603
Defining Validation, Editing, and Estimation (VEE) Options....605
Exception Types....605
Understanding Exception Types....605
Configuring Exception Types....605
VEE Groups....606
Understanding VEE Groups....606
Configuring VEE Groups....606
VEE Rules....606

Understanding VEE Rules ... 607
Configuring VEE Rules ... 607
Validation VEE Rules ... 611
Consecutive Interval Check ... 611
Duplicate IMD Check ... 614
Dynamic Comparison Validation ... 614
Ensure IMD Exists for Sibling MCs ... 616
Final Measurement Replacement ... 617
High/Low Check ... 620
Inactive Measurement Check ... 621
Interval Size Validation ... 623
Interval Spike Check ... 624
Multiplier Check ... 624
Negative Consumption Check ... 624
Prolonged Estimation Check ... 625
Raise Missing Quantity Exception ... 626
Sum Check ... 626
Unit of Measure Check ... 626
Zero Consumption Check ... 627
Estimation VEE Rules ... 627
Interval Adjustment From Scalar ... 627
Interval Averaging Estimation ... 628
Interval Create Estimation IMD for Gap ... 628
Interval Interpolation Estimation ... 630
Interval Profile Estimation ... 630
Interval Proxy Day Estimation ... 631
Scalar Calculation From Interval ... 631
Scalar Estimation ... 631
Scalar Profile Estimation ... 632
Scalar Proration ... 632
Subtractive Interval Adjustment Rule ... 633
Decision-Making VEE Rules ... 633
Exception Handler ... 633
Execute VEE Group ... 634
Successful Termination ... 634
VEE Group Matrix (Factor) ... 634
Measuring Component Statistics ... 635
Understanding Measuring Component Statistics ... 635
Configuring Measuring Component Statistics ... 636
Defining Usage Options ... 638
Usage Subscription Types ... 638
Understanding Usage Subscription Types ... 638
Configuring Usage Subscription Types ... 638
Integrating Usage Transactions ... 639
Requesting Usage Transactions from External Systems ... 639
Processing and Sending Usage Transactions to External Systems ... 639
Generating Usage Transactions in Oracle Utilities Meter Data Management ... 640
Usage Transaction Exception Types ... 641
Understanding Usage Transaction Exception Types ... 641
Configuring Usage Transaction Exception Types ... 641
Usage Calculation Groups ... 641
Understanding Usage Calculation Groups ... 641
Configuring Usage Calculation Groups ... 642
Usage Calculation Rules ... 642
Understanding Usage Calculation Rules ... 642
Configuring Usage Calculation Rules ... 644
Pre-Calculation Usage Calculation Rules ... 648
Alignment and Delay Usage Calculation Rule ... 648
Check Existence of Installed Device ... 648
Calculation Usage Calculation Rules ... 648
Apply Math (Interval Data) ... 649
Daily Scalar Usage Calculation Rule ... 650
Get Interval Data ... 650
Get Item Counts and Consumption ... 650

Get Scalar Details....651
Get Subtractive Interval Details....651
Get TOU Mapped Usage....652
Interval Tier Calculation....652
Profile Accumulation....652
Profile Service Quantity....653
Round and Adjust Usage....653
Vector and Service Quantity Math....654
Post-Calculation Usage Calculation Rules....654
Usage High/Low Rule....654
Validate Against Tolerance....655
Business Flag Hold....655
Decision-Making Usage Calculation Rules....655
Execute Usage Calculation Group....655
Exception Handler....656
Detailed Configuration Examples....656
Usage Calculation Types....657
Understanding Usage Calculation Types....658
Configuring Usage Calculation Types....658
Usage Subscription Quantity Types....658
Understanding Usage Subscription Quantity Types....658
Configuring Usage Subscription Quantity Types....659
Dynamic Option Types....659
Understanding Dynamic Option Types....659
Configuring Dynamic Option Types....659
Contact Types....660
Understanding Contact Types....660
Configuring Contact Types....660
Bill Cycle....660
Understanding Bill Cycle....660
Configuring Bill Cycle....660
Defining Time of Use Mapping Options....661
Time of Use Groups....661
Understanding Time of Use Groups....661
Configuring Time of Use Groups....661
Time of Use Map Templates....662
Understanding Time of Use Map Templates....662
Configuring Time of Use Map Templates....663
Time of Use Map Types....663
Understanding Time of Use Map Types....663
Configuring Time of Use Map Types....664
Defining Communications Options....665
Device Event Types....665
Understanding Device Event Types....665
Configuring Device Event Types....667
Activity Types....667
Understanding Activity Types....667
Configuring Activity Types....669
Communication Types....670
Understanding Communication Types....670
Configuring Communication Types....670
Service Task Types....670
Understanding Service Task Types....670
Configuring Service Task Types....671
Smart Grid Gateway Adapters....672
Smart Grid Gateway Adapters and On-Premises Implementations....672
Configuring Smart Grid Gateway Adapters....673
Itron OpenWay....673
Itron OpenWay Adapter Processing....673
Configuring an Itron OpenWay Head-End System....687
Configuring Itron OpenWay Extendable Lookups....693
Using the Itron OpenWay Test Harness....695
Landis+Gyr....695

Landis+Gyr Adapter Processing..... 696
Configuring a Landis+Gyr Head-End System.....709
Configuring Landis+Gyr Extendable Lookups.....715
Using the Landis+Gyr Test Harness..... 716
Landis+Gyr Interval Data Mapping..... 717
MV90 for Itron.....722
MV90 Adapter Processing..... 723
Configuring an MV90 Head-End System.....726
Configuring MV90 Extendable Lookups.....729
Networked Energy Services..... 730
Networked Energy Services Adapter Processing..... 730
Configuring a Networked Energy Services Head-End System.....744
Configuring Networked Energy Services Extendable Lookups..... 750
Configuring NES Usage and Event Extract Processing.....752
Sensus..... 756
Sensus Adapter Processing.....756
Configuring a Sensus Head-End System..... 767
Configuring Sensus Extendable Lookups.....772
Using the Sensus Test Harness.....773
Silver Spring Networks..... 773
Silver Spring Networks Adapter Processing.....773
Configuring a Silver Spring Networks Head-End System..... 784
Configuring Silver Spring Networks Extendable Lookups..... 790
Using the Silver Spring Networks Test Harness..... 791
Using Smart Grid Gateway Test Harnesses..... 791
Test Harness Design.....792
Locating the WSDL for the Test Harness.....794
Web Services.....794
General Services.....794
Locate Meter Services..... 795
Meter Administration Services..... 797
Meter Attribute Administration Services.....799
Creating a Custom Adapter for Smart Grid Gateway..... 802
Adapter Development Kit Overview..... 802
Adapter Development Kit Processing..... 803
Initial Measurement Data and Device Event Loading..... 803
Device Communication.....808
Oracle Service Bus Processing.....813
OSB Overview.....813
Oracle Utilities Smart Grid Gateway Adapters..... 815
Business Processing Execution Language Processing..... 827
WSDLs, Endpoints, and Messages..... 827
Composite Components.....828
Configuring and Customizing Adapter BPEL Processes.....830
Commands.....832
Working with Enterprise Manager.....841
MultiSpeak Implementation..... 842
Configuring an Adapter Development Kit Head-End System..... 842
Inbound Web Services.....843
Message Senders..... 843
Outbound Message Types.....844
External System..... 844
Service Provider..... 845
Processing Methods.....845
Configuring Endpoint URIs.....846
Configuring Adapter Development Kit Extendable Lookups..... 847
Using the Adapter Development Kit Test Harness..... 849
Locating the WSDL for the Test Harness..... 849
Web Services..... 849
Sample Meters File..... 857
The Adapter Development Kit Native Format..... 858
Adapter Development Kit Native Format Example..... 859
Adapter Development Kit Native Format Schema..... 860
Smart Grid Gateway Adapters and Oracle Utilities Cloud Services..... 878

Payload Processing ... 878
Creating Key Rings and Pairs ... 878
Creating Object Storage Locations ... 879
Creating File Storage Extendable Lookup Values ... 881
Creating Head End Systems ... 882
Configuring Adapter Extendable Lookups ... 882
Creating SGG Payload Processing Extendable Lookup Values ... 883
Payload Handler Classes and Parameters ... 884
Filtering Payloads ... 888
Upload Statistics Activities ... 888
Creating Payload Processing Batch Controls ... 889
Mapping Additional Fields ... 890
Sample Implementation ... 891
Itron OpenWay Payload Processing ... 896
Adapter Development Kit Custom Payload Processing ... 896
Introduction ... 897
Custom Payload Processing Overview ... 897
Payload Processing User Exit Interceptor Scripts ... 898
Configuration Steps ... 898
Sample Implementation ... 899
Smart Meter Commands ... 906
Device Communication Overview ... 906
Smart Meter Command Flows ... 908
Itron OpenWay Command Flows ... 908
Landis+Gyr Command Flows ... 909
Sensus Command Flows ... 910
Silver Spring Networks Command Flows ... 911
Sending Outbound Communications ... 912
Creating Outbound Message Types ... 913
Creating Message Senders ... 915
Creating an External System ... 916
Configuring Processing Methods ... 919
Receiving Inbound Communications ... 923
Inbound Web Services ... 923
Adapter Development Kit - Creating Custom Commands ... 924
Overview ... 924
Synchronous Commands ... 929
Asynchronous Commands ... 933
Cloud Service Test Harness ... 947
Adapter-Specific Test Harness Commands ... 949
Master Configurations ... 955
Defining Service Order Management Options ... 957
Understanding Service Order Management ... 957
Service Order Activities ... 958
Understanding Service Order Activities ... 958
Service Order Activity Processing ... 959
Service Order Activity Algorithm Types ... 961
Understanding Service Order Activity Types ... 966
Configuring Service Order Activity Types ... 967
Service Order Field Activities ... 967
Understanding Service Order Field Activities ... 967
How Do Service Order Field Activities Work? ... 968
Service Order Field Activity Processing ... 969
Service Order Field Activity Communication ... 973
Unrelated Pickup Orders ... 976
Understanding Service Order Field Activity Types ... 977
Configuring Service Order Field Activity Types ... 977
Field Task Types ... 977
Understanding Field Task Types ... 978
Configuring Field Task Types ... 978
Service Order Management External Applications ... 979
Defining Settlement Options ... 981
Settlement Configuration Overview ... 981
Settlement Subscription Types ... 981

Understanding Settlement Subscription Types....981
Configuring Settlement Subscription Types....982
Settlement Transaction Exception Types....982
Understanding Settlement Transaction Exception Types....982
Configuring Settlement Transaction Exception Types....982
Settlement Units....983
Understanding Settlement Units....983
Configuring Settlement Units....983
Settlement Data Snapshot Types....984
Attribute Groups....984
Understanding Attribute Groups....984
Configuring Attribute Groups....985
Attribute Data Snapshot Types....985
Understanding Attribute Data Snapshot Types....985
Configuring Attribute Data Snapshot Types....985
Measurement Data Snapshot Types....985
Understanding Measurement Data Snapshot Types....985
Configuring Measurement Data Snapshot Types....986
Settlement Calculations....986
Settlement Calculation Groups....986
Understanding Settlement Calculation Groups....986
Understanding Bulk Settlement Calculation Groups....987
Configuring Settlement Calculation Groups....987
Configuring Bulk Settlement Calculation Groups....987
Settlement Calculation Rules....988
Understanding Settlement Calculation Rules....988
Configuring Settlement Calculation Rules....988
Calculation Rules....990
Settlement Item Types....991
Understanding Settlement Item Types....991
Configuring Settlement Item Types....992
Settlement Item Quantity Types....992
Understanding Settlement Item Quantity Types....992
Configuring Settlement Item Quantity Types....992
Market Contract Types....992
Understanding Market Contract Types....993
Configuring Market Contract Types....993
Market Products....993
Market Product Sets....993
Understanding Market Product Sets....993
Configuring Market Product Sets....993
Market Product Types....994
Understanding Market Product Types....994
Configuring Market Product Types....994
Settlement Module Configuration....995
Additional Independent Modules....996
Aggregation....996
Standard Aggregation....996
Configuring an Out-of-the-box Aggregation....996
Understanding an Example Out-of-the-box Aggregation....996
Creating a New Custom Aggregation....997
Dynamic Aggregation....998
Dynamic Aggregation Configuration Overview....998
Aggregation Master Configuration....998
Data Sources....999
Understanding Data Sources....999
Configuring Data Sources....1000
Configuring Data Source Template SQL....1000
Dynamic Aggregation Measuring Component Types....1000
Understanding Dynamic Aggregation Measuring Component Types....1001
Configuring Dynamic Aggregation Measuring Component Types....1001
Aggregation Groups....1001
Understanding Aggregation Groups....1001
Configuring Aggregation Groups....1002

Measuring Component Sets....1003
Understanding Measuring Component Sets....1003
Configuring Measuring Component Sets....1004
Configuring Aggregation Criteria Source Types....1004
Configuration Step-by-Step....1005
Extending Dynamic Aggregation....1006
Consumption Synchronization....1007
Configuring Consumption Synchronization....1007
Dashboards....1011
Configuring the MDM Operational Dashboard....1011
Configuring the Service Order Operational Dashboard....1011
Configuring the Service Order Trends Dashboard....1011
Data Extracts....1012
Configurable Consumption Extracts....1012
Maintenance Object Data Export....1017
Enabling Maintenance Objects for Generalized Data Export....1017
Measurement Data Export....1017
Consumption Extract Type....1022
Understanding Consumption Extract Type....1022
Configuring Consumption Extract Type....1022
Analytics Configuration....1023
Information Lifecycle Management (ILM)....1023
Understanding Information Lifecycle Management (ILM)....1023
Configuring Information Lifecycle Management (ILM)....1029
Measurement Reprocessing....1031
Configuring Measurement Reprocessing....1031
Outage Storm Mode....1031
Configuring Outage Storm Mode....1031
Detailed Examples of Outage Storm Mode....1033
Integrations....1036
External Applications....1036
Understanding External Applications....1036
Configuring External Applications....1037
Business Flags....1037
Understanding Business Flags....1037
Configuring Business Flags....1038
Oracle Utilities Customer Care and Billing....1038
Overview....1038
Configuring Master Data Synchronization....1039
Configuring the Bill Determinant Interface....1045
Oracle Utilities Operational Device Management....1047
Overview....1047
Configuring Master Data Synchronization....1048
Oracle Utilities Analytics....1049
Overview....1049
Configuring Extracts....1049
Oracle DataRaker....1050
Business Flag Integration....1050
Configuring Business Flag Integration....1050
Usage and Event Integration....1051
Configuring DataRaker Usage and Event Integration....1052
Oracle Utilities Network Management System....1053
Overview....1053
Configuring Device Event Notification Suppression....1054
Oracle Utilities Customer Self Service....1054
Overview....1054
Configuring Rate Compare Usage Adjustment Profiles....1056
Oracle Utilities Dataconnect / Opower....1060
Overview....1060
Consumption Extract....1061
Master Data Extract....1064
Extract Flat File Formats....1066
Message Formats....1067
Data Import - Message Driven Bean Configuration....1067

Overview....1067
JMS Configuration....1067
Message Driven Bean Configuration....1068
Notification Queue Configuration....1070
Creating Application Links in External Applications....1070
Data Conversion and Migration....1072
Data Migration Scenarios....1073
Data Migration Processes....1077
Customer Data Migration....1078
Meter Data Migration....1082
Measurement Data Migration....1085
Migrating Data for On-Premises Implementations....1087
Migrating Master and Transaction Test Data....1087

Chapter 1

Framework Administrative User Guide

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

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 addition 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. Refer to Sending Email for more information
Address Geocoding	Optional	Algorithms of this type use Oracle Locator to retrieve latitude and longitude coordinates using address information.
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 values based on the new value of the field that was changed. Refer to Global Context Overview for more information.
Guaranteed Delivery	Optional	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.
Ldap Import	Optional	Algorithms of this type are called for operations on users, groups, and group memberships after they have been processed.
Ldap Import Preprocess	Optional	Algorithms of this type are called to preprocess data retrieved from LDAP.
Next To Do Assignment	Optional	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.
Reporting Tool	Optional	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.
SMS Receive	Optional	This type of algorithm is used to provide SMS receive service. Only one algorithm of this type should be plugged in.
SMS Send	Optional	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

System Event	Optional / Required	Description
		implementation. Only one algorithm of this type should be plugged in.
To Do Information	Optional	We use the term To Do information to describe the basic information that appears throughout the system to describe a To Do entry. Plug an algorithm into this spot to override the system default "To Do information".
To Do Pre-creation	Optional	These types of algorithms are called when a To Do entry is being added. They are typically used to set up additional information for the To Do, such as characteristics. They may also set a flag to indicate that the To Do entry should be suppressed. Algorithms plugged in to this spot will be overridden by pre-creation algorithms configured on the To Do Type, if applicable.

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 **Product Status** is used to indicate if a product installed by default in the version you are logged into is **Active** or **Inactive**.

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, **F1-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.

NOTE: The language pack updates all language entries for base owned system data. If your implementation updates base owned labels and descriptions prior to applying the language pack, they will be overwritten. Note that most user facing labels and messages support defining an Override Label or Override Description. This information is not updated by the base product and should be utilized if your implementation desires a specific label or description.

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**. 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:

Locale	Comments
en_US	American English
en_AU	Australian English
pt_BR	Brazilian Portuguese
fr_FR_EURO	European French
ja_JP	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 https://docs.oracle.com/javase/7/docs/api/java/text/Collator.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.

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 indicating that an address field is **Optional**, you make that field available everywhere addresses for this country are used in the system. You can enter your own descriptions for the labels. These labels will appear wherever addresses are maintained in the system.

NOTE: Your specific product may also add the ability to mark an address field as **Required**. If that is available, then the product is also supplying appropriate validation in all places where a user defines an address.

NOTE: For any country where the **State** is enabled, 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.

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.

Defining Time Zones

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

Designing 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.

The following sections describe concepts and configuration topics related to managing 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.

The Base Time Zone

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.

Standard vs. Legal Time

The term **Legal Time** refers to the actual clock time as may be impacted by daylight saving (also called 'local time'). This time line has a gap hour on entry to the daylight saving period and a duplicate hour on exit from it.

The term **Standard Time** refers to a time line without any daylight saving shifts applied to it. This virtual time line has no gap hour on entry to the daylight saving period nor a duplicate hour on exit, i.e. it is continuous and non-ambiguous. As a general rule, it is recommended that all time sensitive data is stored in the standard 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 time zone or daylight saving shifting of data that may be stored in different time zones.

Depending on your specific product, entities may have their date/time information stored in either of the following options:

- Standard time of the base time zone (also called 'physical time').
- Standard time of another time zone related to the entity's geographical location (also called 'logical time').
- Legal time of the base time zone.

The metadata definition of a table specifies whether any of its date/time fields are stored in standard time or not. While typically all date/time fields of a table that supports standard time would be stored in the same way the system allows each field to specify its own option. The metadata definition of such field specifies whether it is base time zone standard time (**Physical Standard Time**), standard time of some other time zone associated with the entity (**Legal Standard Time**) or stored in base time zone legal time (**Referenced Time Zone**).

Storage vs. Display

Regardless of the time zone option used for storing the data in standard time, all date/time information is entered and displayed on the user interface in the legal time of its respective time zone. If a field is defined to be stored in the standard time of the base or other time zone, the user interface is designed to accept the data in the legal time of the respective time zone and shift it to the standard time of the same time zone before saving the data and performs the reverse shift from standard time back to legal time before displaying the data to the user.

It is important to understand that time zone and daylight saving time conversions to and from standard time are performed behind the scenes as part of the user interface layer. Once data reaches the server it is already in its storage time option. The user should not be aware of these time conversions.

Time shifting between standard and legal times does not happen automatically. When dealing with fields stored in standard time, their data entry and display need to explicitly include time shifting features as part of their user interface and zone configuration.

Date / Time Schema Elements

When defining date / time fields in a business object 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).

By default data stored in standard time is displayed in the corresponding time zone's legal time. Additional schema attributes can be used to indicate if the display of the time should be shifted to the legal time of a different time zone. For example, 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 Nodes and Attributes- Standard Time Considerations for more information.

Exchanging Date / Time Information With External Systems

Date/time information should be exchanged between systems in standard XSD format as it includes a reference to the time zone of the specified time as an offset from Coordinated Universal Time (UTC). The system automatically converts date/time elements in an inbound message from XSD to internal format and vice versa for an outbound message. The latter is controlled by an explicit date/time format setting on the External System record for the outbound message type.

The offset is determined as followed and it is based on the time zone associated with the element as explicitly defined by the respective inbound or outbound schema:

- Elements defined as stored in the standard time of the base or other time zone would always have the same offset throughout the year as this time line is never shifted for daylight saving. The offset would be the standard offset of that time zone from UTC, i.e. not during daylight saving time.
- Elements stored in the legal time of the base time zone would have the standard offset of the base time zone when the date/time value is outside the daylight saving period and the shifted offset when it is inside that period of time.

User Time Zone

If your company does business in multiple time zones, the user record of each user may reference the time zone of their location. This time zone provides additional information about the location of the user and as such may be used by specific business rules that involve time zone settings.

> **NOTE:** The time zone defined on the user record is not used to automatically display date/time information in the user's time zone. By default, the system displays date/time fields that are stored in standard time in the legal time of their respective time zone and not the user's time zone.

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.

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.

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**.

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.

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.

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** and **Description** to identify the profile.

Enter a **Date Format**. This affects how users view dates and how entered dates are parsed. The following table highlights standard supported date mnemonics and what is displayed at runtime.

Mnemonic	Comments
dd	Day of the month.
d	Day of the month, suppressing the leading 0.
MM	Month number.
M	Month number, suppressing the leading 0.
yyyy	The 4-digit year.
yy	The 2-digit year.
y	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.

Examples:

Configuration Format	Sample Output
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 master configuration. 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. The letter 'H' is used for dates that fall within the *Heisei* era. The letter 'R' is used for dates that fall within the *Reiwa* 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**. The following table highlights standard supported date mnemonics.

Mnemonic	Comments
hh	The hour 1-12.
h	The hour 1-12, suppressing the leading 0.
HH	The hour 0-23.
H	The hour 0-23, suppressing the leading 0.
KK	The hour 0-11.
K	The hour 0-11, suppressing the leading 0.
kk	The hour 1-24.
k	The hour 1-24, suppressing the leading 0.
mm	Minutes.
m	Minutes, suppressing the leading 0.
ss	Seconds.

Mnemonic	Comments
s	Seconds, suppressing the leading 0.
a	Indicates to include **am** or **pm**. This is only needed for 12 hour formats, not 24 hour formats. (**hh**, **h**, **KK**, **K**). If an **am** or **pm** is not entered, it defaults to **am**.

Examples:

Configuration Format	Sample Output
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 as follows:

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

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 Mapping**master 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.

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.

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.**

Use the **Feature Configuration Query** portal to search for an existing feature configuration. Once a feature configuration is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Overview / Initial Setup

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.

Implementations may configure the business objects to limit the ones that are visible to users if desired.

- If the master configuration does not apply to your implementation, navigate to the business object for this master configuration and update the Instance Control to **Do not allow new instances**. This ensures that the business object will not be visible to any user for the implementation.
- If a given master configuration record is only relevant for certain users in the system, application security may be used to limit the records visible by a given user. By default the base delivered business objects typically reference the Master Configuration MO application service. This may be overridden to link custom application services to the master configuration BOs to provide more granular security. Users will only see the master configurations that they have security for.

Configuration

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.

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 transactions or explicit services using an application service. The following points highlight what may be secured.

- The following points highlight security related to viewing and modifying individual records 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. Note that the application service for the maintenance object is defined on its related service program.
  - 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 (stand-alone) 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. Note that the application service for the portal is defined on its related service program, which is derived via its navigation option and navigation key.
  - Menu items may define an application service / access mode. Typically the security supplied for portals and maintenance objects provides enough granularity to suppress menu items that a user does not have access to. Linking an explicit application service / access mode will further suppress the menu item under one of 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 menu option, link the object's application service and the Add access mode to the Add menu item.

NOTE: The base product does not typically provide menu items with application services configured. Implementations may add this configuration if one of the above scenarios exist.

- 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 is delivered with 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.
- For portal based pages, individual elements may be shown or hidden based on security using the **oraHasSecurity** function in a UI map's HTML or in UI Hints in a schema. Refer to Check User's Security Access for more information.
- 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. In addition, implementations may override the configured application service if desired.
- Batch controls define an application service which provides the ability to secure submission of individual batch processes. The application service must include an access mode of **Execute**. The base product batch controls are supplied with appropriate application services. These services will typically have an ID that matches the batch control ID.
- Report Definition records define an application service. The application service must include an access mode of **Submit / View Report**.

- The following objects are securable but are typically executed via internal processes. The security is provided to ensure that any access to the objects from an external source is secured.
  - 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.

- Scripts may also check security within an Edit Data step type if there is some functionality that should or should not execute based on a user's security access. The Edit Data step would include a call to the business service **F1-CheckApplicationSecurity**
- Service Programs define an application service. As mentioned above, for Portals and Maintenance Objects, their application service is taken from the related service program. In base, specific application services are released for each of these types of service programs. All other service programs are typically delivered with a default application service, which may be overridden by an implementation. Note that for service programs linked to a Business Service, the application service on the business service takes precedence when invoking the business service.

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.

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. There are two different types of encryption described in the sections below. System encryption refers to columns in the system identified by the product to use encryption. Application encryptions refers to the ability for an implementation to configure fields and element that should be encrypted in the database.

"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 fields on the user interface that may be stored as plain text in the database. In addition, there are cases where encrypted fields are shown to the user interface using masked values rather than the encrypted value.

The following sections provide more information about each feature.

System Encryption

The system automatically encrypts certain fields captured in various option tables or context tables. This is mainly used for passwords. For example, passwords captured in Message Sender context or password

In addition, batch control supports configuring a security option for parameters that capture sensitive information, such as a password. Refer to Defining Batch Controls for more information.

It is also possible to enable system encryption using the characteristic type **F1-PWD**. However, the maintenance object must include specific code to enable system encryption for characteristics of this type. In Oracle Utilities Application Framework, the only maintenance object that supports this is extendable lookup. Refer to Extendable Lookup Advanced Topics for more information.

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 (using the algorithm entity value of **Feature Configuration - Data Masking**) 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 **F1-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.

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 and 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 in 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.

Application Encryption

The functionality described in this section allows implementations to configure fields to encrypt when storing it in the database. This functionality is mutually exclusive from the User Interface Masking functionality described in the previous section. This feature supports encrypting specific elements stored within a CLOB or XML column.

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 Feature Configuration - 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.

A combination of framework configuration and configuration in your edge product is required for row level security. The following points describe the configuration:

- For each record that should be secured, associate it with an **Access Group**. Note that if your edge product supports row level security, that product is providing a link between the secure-able record and Access Group. Your access groups may be granular and only referenced by one secured record or they may be more broad and be referenced by multiple secured records that require the same type of security restriction.
- To define which users have access to the secured records, you define a **Data Access Role**. For each data access role, define which Access Groups the role has security clearance for. An access group may be linked to one or more data access roles. In addition, define the **Users** that have access rights to these secured records. 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 secured records in all the referenced access groups. A user may belong to many data access roles.

If your edge product supports row level security, it will include logic in the appropriate areas of the system to limit the secured rows that a user may view or maintain based on this configuration. For example, in Oracle Utilities Customer Care and Billing, throughout the system users are only able to view and maintain information about an account and any of its detail if the user is in a Data Access Role for the account's Access Group (or the account is not linked to an Access Group).

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

Defining Application Services

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.

Application Service - Application Security

Use the Application Security portal to view current information about the relationships for the application service and to set up / modify 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.

Secured Objects zone. This zone displays the object (or objects) that are secured by this application service. Refer to Application Security for details about the types of objects that may be secured.

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.

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. Use the **App Service Description** search to restrict the application services displayed in the grid.

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.

Please note the following with respect to maintaining application services linked to the user group:

- 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 delete icon to remove the application service from the user group.

FASTPATH: Refer to User Group Services Management for a more convenient, faster method to add or remove application services to or from a user group in bulk.

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.

FASTPATH: Refer to User Group Services Management for a more convenient, faster method to add or remove application services to or from a user group in bulk.

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** 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 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.

User Group Services Management

This portal allows a security administrator to more quickly manage the application services and access modes linked to a user group.

Open this page using **Admin** > **Security** > **User Group Services Management**.

This section describes the available zones on this page.

The **User Groups** zone provides a list of user groups in the system. Broadcast the desired user group to enable the subsequent zones.

The **Application Services Linked** zone displays a row for each application service and access mode combination currently linked to the user group. The zone includes a filter area to limit the list based on criteria. A user can select one or more rows to remove the selection from the user group.

The **Application Services Not Linked** zone displays a row for each application service and access mode combination not currently linked to the user group. The zone includes a filter area to limit the list based on criteria. A user can select one or more rows to add the selection to the user group and is prompted for the expiration date.

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.

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

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.

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.

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.

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.

Data Privacy

Personally Identifiable Information (PII) is any information that identifies or could be used in combination with other information to identify an individual. Many countries have data privacy laws governing the use of such information. A key aspect of these laws is an organization’s obligation to erase personal information when there is no compelling reason to retain it. This obligation can be fulfilled either by deleting data or altering the data in such a way that it is no longer possible to associate the data with the individual.

Organizations are likely to have other legal obligations that impact how long they retain personal data. For example, there could be financial audit requirements that oblige the organization to retain relevant information for a fixed number of years after termination of a contract.

Object Erasure is designed to address right to erasure issues, with a combination of configuration and processes that may be extended to implement the appropriate policies for the PII within your data.

The Approach to Implementing Object Erasure

This section describes the product approach to implementing object erasure for its maintenance objects (MOs).

NOTE: The approach outlined is intended for managing erasure of master data objects. The assumption is that Information Lifecycle Management will be used for archiving any related transactional data.

Various events in the system may signal the need to erase data at some future date. For example, closing an account may indicate the need to erase both the account and person data once there is no longer a need to maintain related financial details. There may also be events that re-establish the business relationship with a person and signal that data marked for erasure should now be retained.

The Object Erasure Schedule is used to capture key details of objects with data that needs to be erased. Records in the schedule are monitored periodically to determine if the date for erasure has been reached and if so, execute the erasure logic. The system will create and maintain records in the schedule when specific events occur for an MO that is marked for erasure.

The system provides the ability to mark an MO as eligible for erasure via a number of MO configuration entries. Included in these is an option that defines the business object for the MO’s erasure schedule records. In addition to defining the erasure record elements, this BO defines the monitoring process and the specific erasure logic for the MO.

The following sections provide more information about configuring and managing object erasure.

Configuring a Maintenance Object for Erasure

There are three types of maintenance object configuration entries that define how to manage erasure:

- An option to define the business object for erasure schedule records.
- An option to define the period between the date on which the need for erasure is detected and the date on which erasure should occur. The common logic provided by the system to add or update erasure records uses this option to determine the erasure date.
- One or more entries in the MO algorithm collection for the algorithms that determine whether a record should be scheduled for erasure, according to the current state of the record, and create or update the erasure schedule entries.

Some MOs that would commonly be regarded as eligible for erasure will be installed with a base Erasure BO already configured. The expectation is that the corresponding Erasure Period option and the algorithm for the Manage Erasure Schedule plug-in spot will be set up by your implementation with the appropriate entries for your organization's business rules.

> **NOTE:** The presence of an Erasure Period option is the signal to the system that your implementation has enabled an object for erasure.

It may not be necessary for all master data objects that contain personally identifiable information to manage their own erasure schedule. For example, the event which triggers erasure for a person record may also trigger the creation of erasure schedule entries for the person's other master data records. In this case, only the person maintenance object will need to be configured with a Manage Erasure algorithm (or algorithms). Refer to your product specific documentation for more information on the recommended approaches for your product's master data objects.

Manage Erasure Schedule Algorithm

Algorithms of this type are responsible for determining a record's erasure status and creating or maintaining an entry in the Object Erasure Schedule for the record. They are triggered when certain events occur in the system for maintenance objects that are eligible for erasure. Multiple algorithms can be configured for the plug-in spot.

Depending on the determination made by the algorithm (or algorithms), a number of different actions may need to be taken. For example:

- A new schedule record may need to be added
- An existing schedule record may need to be updated with a new erasure date
- A pending record may need to be discarded if erasure is no longer applicable
- A discarded record may need to be reactivated if erasure now applies again

The base business service **F1-ManageErasureSchedule** performs the common logic to handle the possible actions and is recommended for use by these types of algorithms. The service can also optionally create a log entry for information purposes. Refer to the description of the business service for full details.

In certain circumstances, if one record is scheduled for erasure, other related records may need to be evaluated. The base business service **F1-ManageErasureScheduleDriver** performs the logic to execute an MO's Manage Erasure Schedule algorithms. It can be used to manage any related records from within another algorithm.

This type of algorithm is plugged into the Maintenance Object — Algorithm collection.

Your product may supply algorithms to manage the schedule for base MOs that are regarded as eligible for erasure. If your organization has special business rules that are evaluated to determine the erasure schedule for an MO, a custom algorithm can be created and applied by the implementation team.

Monitoring the Schedule and Performing Erasure

The erasure schedule is managed using the lifecycle of the erasure record's business object. A deferred monitor process is used to periodically check for records that are due for erasure and transition them from the 'pending erasure' state to the 'erased' state. An enter plug in on the 'erased' state is responsible for performing the erasure logic.

The system provides a 'root' business object for the object erasure schedule (**F1-ErasureScheduleRoot**) which defines the lifecycle that erasure schedule business objects should follow. The system also provides a monitor process (**F1-OESMN**) which is configured on the 'pending erasure' state of the 'root' business object. This process is configured to monitor object erasure schedule records whose erasure date is on or before the batch business date.

Maintenance objects that are eligible for erasure should be configured with an erasure schedule business object that uses the 'root' business object as its parent. The algorithm that performs the erasure processing applicable to that maintenance object must be plugged in on the 'erased' state of the 'child' business object.

The system provides an erasure schedule business object (**F1-ErasureScheduleCommon**) which is designed to erase an object by deleting the main record and any child records. Your product may supply additional erasure schedule business objects for certain use cases. Refer to your product specific documentation for more information.

Erasing User Information By Obfuscation

Some implementations consider the base User object to have personally identifiable information. Erasing user information by deleting the records is not advisable as it can cause referential integrity problems. The recommended approach is to obfuscate the data instead.

The system provides a user erasure business object (**F1-ErasureScheduleUser**) which is designed to support obfuscating a user's identifiable information by removing it or replacing it with a non-identifiable value if the field is required.

If the User maintenance object is configured to be eligible for erasure in your implementation, the manage erasure schedule algorithm for the record will be invoked whenever the **Enable** flag is changed for a record on the User maintenance page. The system provides a base algorithm (**F1-OBJERSUSR**) specifically for the User MO. This algorithm will add an entry to the object erasure schedule if the user record is disabled or deactivate an existing erasure schedule record if a disabled user record is enabled again.

Viewing an Object's Erasure Status

The system provides a portal for viewing and editing an erasure schedule record. The portal is accessed from the **Object Erasure Schedule** dashboard zone. This is a context sensitive zone which only appears when accessing a record for a maintenance object that is configured to be eligible for erasure. It displays the information for the erasure schedule entry for the record in context, if applicable.

To add the zone to an eligible maintenance object:

- Navigate to Context Sensitive Zone and search for the navigation key for the maintenance object.
- Add the Object Erasure Schedule zone **F1-OBJESRSCD** to that navigation key.

Viewing Erasure Configuration

The Erasure Configuration All-in-One portal provides the ability to view the maintenance objects that are marked as eligible for erasure and maintain their erasure configuration entries in a single place.

To view a list of the eligible MOs, open **Admin** > **Security** > **Erasure Configuration.**

Press the edit icon to open a window in which the configuration values for that row can be changed.

Refer to The Approach to Implementing Object Erasure for an overview of object erasure configuration.

Archiving the Object Erasure Schedule

The system provides the ability to archive older records in the object erasure schedule using Information Lifecycle Management. The object erasure schedule maintenance object is configured with a base ILM eligibility algorithm and an ILM crawler batch control.

The base object erasure schedule BO is configured to set the ILM archive switch when an erasure schedule record enters a final state and to reset it if the record becomes active again.

Refer to The Approach to Implementing Information Lifecycle Management for more information on archiving functionality.

Cryptography Keys

Often when communicating information with an external system, cryptography keys are used to exchange encrypted information or confirm that the two communicating parties recognize each other and the information being provided.

The following sections include information about the functionality provided to support this functionality.

Understanding Key Rings

Cryptography keys may be used to provide a signature to a request so that the system recognizes that the request comes from a trusted party. Keys may also be used to encrypt or decrypt files shared between two parties.

In this release, support for a Signature type of key is provided to be able to access files stored in Oracle Cloud Object Storage. This support is only applicable for cloud customers.

The product supports the ability for the following:

- A pair of keys - public and private. For keys generated in the system, the private key is stored in an appropriate "secret store" and the public key is available to copy and share with a third party.
- Key rotation. For increased security, a new key pair should be generated periodically.

The Key Ring object is provided to reference the key pairs that are used over time for a given business use case. Only one key pair may be active at any given time.

The following sections include information about the functionality provided to support this functionality.

Signature Keys

Oracle Cloud Object Storage is used by cloud customers to store any files that are needed by the system. Certain processes in the system may need to write files to or read files from cloud object storage. Cloud object storage has its own method for encrypting files and products using the framework application do not need to define additional configuration for that. However, when communicating with cloud object storage, the application is required to provide a signature key.

Refer to External File Storage for more information about configuring the system to connect to cloud object storage.

The product provides a Signature Key Ring business object that is available for cloud customers. This business object provides the following support

- Generating a key pair, for key rotation
- Viewing the public key, allowing a user to copy it to register it with cloud object storage.
- Activating the new key pair (to be done after registering the new public key)

Defining Key Rings

Refer to Understanding Key Rings for an overview of key ring functionality.

To maintain the key rings applicable to your product or implementation, open **Admin** > **Security** > **Key Ring.**

This is a standard All-in-One portal and includes the standard List and display zones for a statistics control.

The information captured on the key ring depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Key Pairs

The key pair zone provides information about the keys that are associated with the key ring. Actions available on the zone depend on the type of key. Refer to embedded help for more information.

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**.

Use the **Menu Query** portal to search for an existing menu definition. Once a menu is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

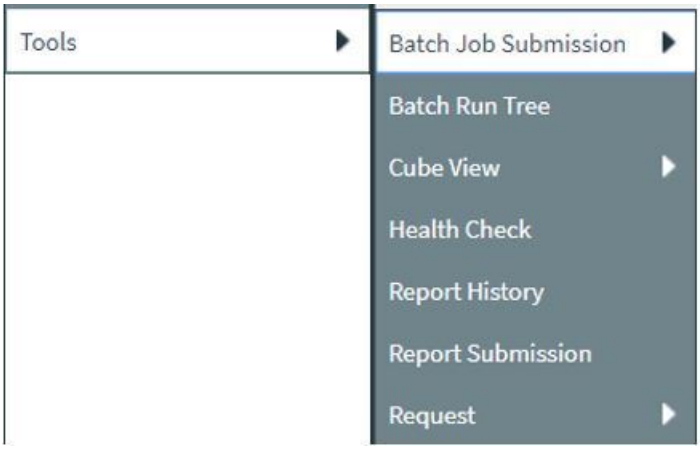
Menu - Menu Items

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 carryover 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.

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-DFLTS) 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 addition 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 Types 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.

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.

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

Use the **Zone Query** portal to search for an existing zone. Once a zone is selected, you are brought to the maintenance portal to view and maintain the selected record.

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, an 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 Zone Parameter Details 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.

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:

For example, your text string could be **revenue.** This would make the word "revenue" appear as red, large and bold. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

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

NOTE: Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.

Zone Parameter Details

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

Zone Visibility Service Script

All zones support a visibility script that is used to determine if the zone should be displayed to the user or not based on conditions. The script may receive input parameters and is expected to return a Boolean value indicating if the zone should be displayed or not. The embedded help for the **Zone Visibility Service Script** parameter provides details related to the syntax.

The following table highlights some service scripts provided by the product that may be used if applicable to your zone's requirements. This is not an exhaustive list of visibility scripts. There may be others that are specific to a given zone.

Script Code	Description	Comments
F1-ShldShwZn	Zone Visibility - Display Zone in Portal	This script simply returns a value of 'true' and is used when the zone should always appear.
F1-CondShwZn	Zone Visibility - Display Zone in Portal Conditionally	This is used when the condition for showing the zone is based on the population of a context value. This is commonly used when one zone in the portal should only appear after a broadcast of a record from another zone in the portal. For an example of a zone the uses this visibility script, refer to **F1-BSFTYPE** .
F1-RwCtShwZn	Zone Visibility - Based on Row Count	This is used when the condition for showing the zone is based on the existence of one or more rows that can be determined using SQL. This script accepts a zone code, user filters 1 through 25 and hidden filters 1 through 10. The script returns an indication of 'true' if at least one row count is returned by the zone. To use this visibility script, a specific data explorer zone must be developed for the specific use case. For an example of a zone the uses this visibility script, refer to **F1-MIGRREQEL** .

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**.

NOTE: If your implementation has been configured to restrict the functions that may be used when defining an SQL then an error is issued at runtime if there are functions found that are not in the whitelist. The whitelist may be viewed using the **View SQL function whitelist** link in the Tips zone on the zone maintenance page.

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. **Numresults** is the number of variations to consider for the string. Valid values are between 1 and 5000. Default is 100. 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.	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
@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

Keyword	Description	Examples
@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 resulted 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**). For this zone type, all SQLs are executed and the zone displays a union of all the results. The parameters follow the description pattern of **Column x for SQL y**. There are some mnemonics that do not make sense to differ within rows of the same column. For example, the column label. For these mnemonics, only the value in SQL 1 is considered for that column. The table below indicates which mnemonics follow this rule.
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**). For this zone type, only one SQL is executed. The **SQL y Condition** parameter may be used to control this. The system will execute the first SQL whose condition is satisfied (or with no condition populated). 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
Common Use Cases

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 also reference the **bs=**, **input=** and

Mnemonic	Description	Valid Values	Comments
			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 and will be enabled for navigation). 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. **NOTE:** To show a foreign key's information but not enable navigation, refer to Common Use Cases for the recommended technique.

Mnemonic	Description	Valid Values	Comments
		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.
		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.	'Business Object Code'	
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.	'Business Service Code'	
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.	'Service Script Code'	
fkref=	Defines the FK reference used to retrieve the column's information when **source=FKREF**. This mnemonic must be used in conjunction with the **input=** mnemonic	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.

Mnemonic	Description	Valid Values	Comments
	to define how information is sent to the FK reference to build the information.		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=	Defines the formula to use when **source=FORMULA**. Examples: • **formula=C1*.90/C2** • **formula=(C1/C2)*100**	The formula can contain numeric constants, operators and column references.	For column references, use the format Cx where x represents the column number. Refer to Expression Parser for information about the functions supported.
setfunc=	Defines the function to apply to the rows of a given column when **source=SETFUNC**.	function(Cx)	Where Cx represents a column whose rows should have the function applied and the function is one of the following: • **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=	This is used to define one or more input fields and values passed to business objects, business services, service scripts, and FK references. The syntax is as follows: [ELEMENT_NAME=ELEMENT_REF ELEMENT_NAME=ELEMENT_REF ...] 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 in. The next column indicates the possible values for ELEMENT_REF.	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.
		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: • **input=[USER_ID=C1]**	

Mnemonic	Description	Valid Values	Comments
		• **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.

NOTE: If multiple columns are configured with the same source BO, BS or SS and the same input data, the system caches the output from the first call and reuses the results for subsequent columns.

Formatting Mnemonics

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

NOTE: For the **F1-DE** zone type, zone displays the union of all the different SQLs.

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**	Columns of this type capture a string. This is the default value.
		DATE	Columns of this type capture a date and will be displayed using the user's display profile.
		TIME	Columns of this type capture a time (in database format) and will be displayed using the user's display profile.
		DATE/TIME	Columns of this type capture a date and time (in database format) and will be displayed using the user's display profile.
		MONEY	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. 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. **NOTE:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will use the column definitions in SQL 1 to define the labels.	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.
		'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

Mnemonic	Description	Valid Values	Comments
			with no decimal places (truncated), however any references to the column in subsequent calculations 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.
char=	This mnemonic applies special character(s) to the column's value.	'x[]x'	Where x references the literal value to display and **[ ]** defines the relative position of the characters (before or after the value). 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)'**.
suppress=	This is used to indicate a column should not be displayed. 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. **NOTE:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.	**true**	
		false	This is the default, meaning that you don't need to indicate **suppress=false** at all to indicate that the field should be shown.
suppressSearch=	This is used to indicate a column should not be displayed when the zone is invoked in search mode only. **NOTE:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.	**true**	
		false	This is the default, meaning that you don't need to indicate **suppressSearch=false** at all to indicate that the field should be shown.
suppressExport=	This is used to indicate a column should not be downloaded to Excel.	**true**	

Mnemonic	Description	Valid Values	Comments
	NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.	**false**	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.
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	Where n is a number between 0 and 999. **NOTE:** If there is no available breaking point in the data, the column will be longer than the specified number of pixels. The length of the column's label (which appears in the column's heading) may also make the width wider than specified. In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will set the width based on the column definitions in SQL 1.
color=	This is used to override the column's text color.	A valid HTML "named" color	For example **color=yellow**. **NOTE:** Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.
		A valid RGB color model combination	For example **color=#E0292F** or **color=#CCCCCC**. Note that the **#** is required.
bgcolor=	This is used to override the column's background color.	A valid HTML "named" color	Similar to the **color=** mnemonic.
		A valid RGB color model combination	Similar to the **color=** mnemonic.
order=	Defines the column's default sort order.	**ASC**	Indicates that the order is ascending. This is the default meaning that it is not necessary to indicate **order=ASC**.
		DESC	Indicates that the order is descending.
rowHeader=	Designates the column as a row header for accessibility purposes.	**true**	By default, the first data column of the data explorer results is identified as the row header, for accessibility tools. If the data in the first column does not uniquely identify the row, use this mnemonic to explicitly mark a different column or multiple columns as the row header.

Mnemonic	Description	Valid Values	Comments
			NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will look only at the columns in SQL 1 that have this mnemonic defined and apply it to all results for those columns.

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	This means navigation option code is defined in an earlier column. For example, define **C1** if column 1 defines the navigation option.
		COLUMN_NAME	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**
		'Navigation Option Code'	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	Where Cx represents the value of a previous column. For example, 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.
bpa=	Indicates that a BPA script should be executed with the user clicks the column and indicates the BPA to execute. Note, this mnemonic should be used in conjunction with the **tempstorage=** mnemonic to define the temporary	Cx	Indicates that the BPA script is defined in a previous column.
		COLUMN_NAME	This means the BPA script to execute was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.

Mnemonic	Description	Valid Values	Comments
	storage values that will be initiated when the script is executed. This mnemonic is ignored if **source=FKREF** because the FK reference code defines the hyperlink's destination.	'BPA Script Code'	This means that the BPA script to execute is defined directly.
tempstorage=	This is used to define how temporary storage variables are initiated when the **bpa=** mnemonic is used. 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 field in temporary storage. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.	Cx	Where Cx represents the value of a previous column. For example, 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.
list=	This is used to enable work list capability for this column. You may optionally populate the **listdesc=** mnemonic to override the text that will be placed in the worklist zone.	**true**	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. **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. Also note that when determining which columns should have the worklist icon when building the zone, the system only looks at the configuration for the columns in SQL 1.
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**.

Mnemonic	Description	Valid Values	Comments
listbroadcast=	Indicates that the broadcast information for the column is also to 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.	**true**	Use this setting to turn on the feature.

Common Use Cases

This section provides some common configuration that can be copied and pasted for newly created zones.

Use Case	Code Snippet
Displaying the description of a lookup value that has been retrieved in the SQL statement. The business service checks for an override label.	**source=BS** **bs='F1-GetLookupDescription'** **label=FLD_NAME** **input=[fieldName='FLD_NAME' fieldValue=FLD_NAME]** **output=description**
Display a foreign key with its information but suppressing the navigation. (This is useful when you have a foreign key that is additional information in a zone but would cause confusion to enable navigation. To display foreign key reference with its information and enabled for navigation, use the source of FKREF as described above.) The first example assumes that you know the FK Reference. The second example uses a business service that receives the maintenance object as input.	**source=BS** **bs='F1-GetForeignKeyReference'** **input=[input/fkReference=FK_REF_CD input/fkValue1=BUS_OBJ_CD]** **output=output/infoDescription** **label=DESCR** **source=BS** **bs='F1-GetFKReferenceDetails'** **input=[input/maintenanceObject='F1-EXT LKUP' input/pkValue1=BUS_OBJ_CD input/pkValue2=C1]** **output=output/infoDescription** **label=DESCR**

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.

Mnemonic	Description	Valid Values	Comments
		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 type=ASIS.	**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 **action=NAVIGATION**.	'NAV_OPT_CD'	Enter a reference to a valid navigation option in single quotes.
bpa=	Defines the script to run when the **action=SCRIPT**.	'SCRIPT_CD'	Enter a reference to a valid BPA script in single quotes.
icon=	Indicates the icon to use when **type=ICON**.	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 **type=ASIS**. 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 page or executing a BPA script. The mnemonic supports passing multiple values. 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. 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.	FIELD_NAME	Indicates that the value should be taken from the field with this name from portal context, global context or the page data model. The mnemonic sourceLoc is used for defining the source.
		xpath	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.
		'constant'	Indicates that the value defined in single quotes should be passed.
sourceLoc=	This mnemonic defines the source of the FIELD_ NAME's value in the context mnemonic.	**G**	Indicates that the field's value is retrieved from the global context.

Mnemonic	Description	Valid Values	Comments
	If this mnemonic is left blank, the default behavior is as follows: - 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**	Indicates that the field's value is retrieved from the portal context.
		D	Indicates that the field's value is retrieved from the page data model.
class=	Use this mnemonic to override the look and feel of the link / icon / button using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes. 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.

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=['<A class="oraLink" href="www.google.com">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.	MD 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.	**G**	Indicates that the zone should look for the filter value in global context.
		P	Indicates that the zone should look for the filter value in portal context.
		D	Indicates that the zone should look for the filter value in the page data model.

Mnemonic	Description	Valid Values	Comments
	- 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.		
type=	Defines the visual metaphor used to capture the filter values.	**DATE**	Filters of this type capture a date.
		DATE/TIME	Filters of this type capture a date and time.
		STRING	Filters of this type capture a string
		MONEY	Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.
		NUMBER	Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.
		LOOKUP	Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.
		TABLE	Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.
		CHARTYPE	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.
		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 and in the input area.	MD Field Name	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.
		'text'	Defines the text directly.
cur=	Defines the currency code applied when **type=MONEY**.	Currency Code	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when **type=NUMBER**.	Valid 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.
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 code	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.	XPath	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.

Mnemonic	Description	Valid Values	Comments
		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.	MD 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 if another field has a certain value. The following is an example. **encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,'SSN']**. The WHERE_VALUE may also reference another filter. The following is an 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**	Indicates that the zone should look for the filter value in global context.
		P	Indicates that the zone should look for the filter value in portal context.
		D	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**	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.
		O	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**	Filters of this type capture a date.
		DATE/TIME	Filters of this type capture a date and time.
		STRING	Filters of this type capture a string
		MONEY	Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.
		NUMBER	Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.
		LOOKUP	Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.
		TABLE	Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.
		CHARTYPE	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.
		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.

Mnemonic	Description	Valid Values	Comments
		'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.
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.

Mnemonic	Description	Valid Values	Comments
		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]	This mnemonic is used to populate the list with the appropriate information from each selected row. The mnemonic supports passing multiple values.
	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.
	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.
	Example of a schema:
	`<schema>`
	`  <accountInfo type="list">`
	`  <accountId/>`
	`  <name/>`
	`  <amount/>`
	`  <process/>`
	`  </accountInfo>`
	`</schema>`
	Example of list and context mnemonics.
	list=accountInfo
	context=[accountId=ACCT_ ID name=C2
		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
	amount=P3 process='O']		
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 embedded 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 embedded 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 includes 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.
  - It is recommended to configure the **SQL Display Column** parameter to show the data in the same order as the ORDER BY clause.

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 ASSIGNED_TO>:C3) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and ASSIGNED_TO=:C3 AND TD_ENTRY_ID>:C4))]
[(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 ASSIGNED_TO<:C3) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and ASSIGNED_TO=:C3 AND TD_ENTRY_ID<:C4))]
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 embedded 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 displayed 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. (Obsolete). This information is no longer used by the timeline zone and will be ignored.
  - Icon use for the event. (Obsolete). This information is no longer used by the timeline zone and will be ignored.
  - 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. 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.**

Use the **Portal Query** portal to search for an existing portal. Once a portal is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

  NOTE: Recommendation. It is recommended that zones that have information that is always needed when users first display a portal be set up to be initially collapsed. That way the data in the zone is only built when the user expands the zone. This improves response times.

- **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.

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 a unique value within a given option type.

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.

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 a 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 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.

Use the **Navigation Key Query** portal to search for an existing navigation key. Once a navigation key is selected, you are brought to the maintenance portal to view and maintain the selected record.

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).

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.

Use the **Navigation Option Query** portal to search for an existing navigation option. Once a navigation option is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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

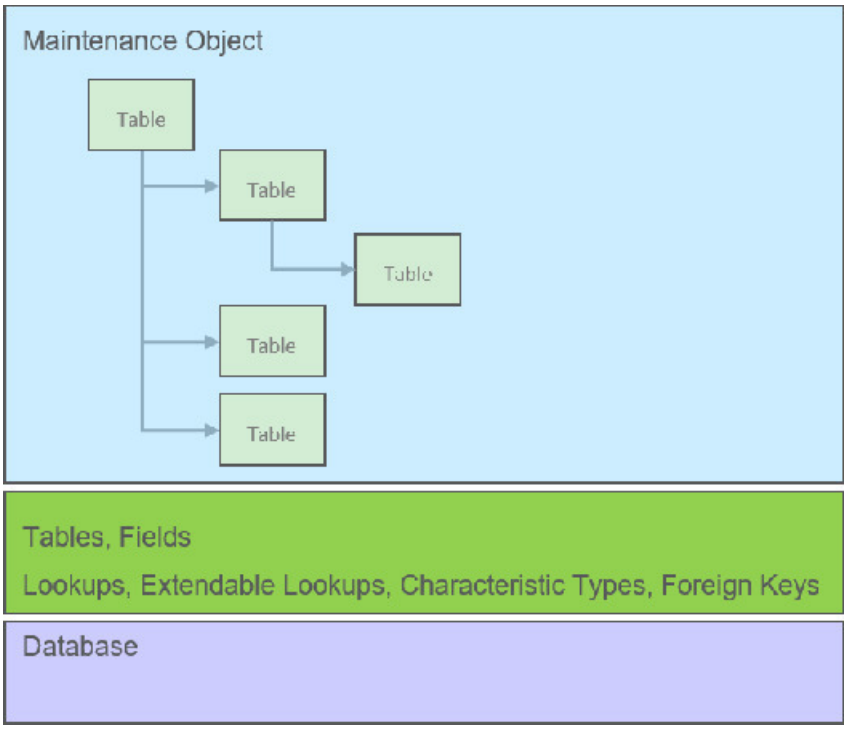
The tree shows every menu item, favorite link, and tree node that references the navigation option. This information is provided to make you aware of the ramifications of changing a navigation option.

Database Options

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

Understanding The Data Model

The application uses meta-data to describe its data model, from plain description of its tables and fields to more complex hierarchies of maintenance objects.



The following points briefly outline key aspects of application's data model as illustrated in this diagram:

- Table and Field records describe all the tables maintained by the application.
- Valid values for certain fields are defined using corresponding Lookup and Extendable Lookup records.
- Characteristic types may be defined to support an extension and of a maintenance object via its characteristics table.
- Maintenance Objects define the physical structure of an entity. Business objects further refine maintenance objects to describe business entities. Refer to The Big Picture of Business Objects for more information.
- Foreign Key References define ways for searching and navigating to the maintenance portal of an entity.

The following sections further discusses concepts and guidelines related to the data model.

Fields

A Field definition describes the metadata needed by the application to support the capture and display of a piece of information. Fields may correspond to physical columns on database tables or represent any type of element used for calculation or presentation purposes.

The definition includes the following key details:

- Whether the field refers to a physical column on a database table or used as a work field.
- The type of data the field holds.
- The label to use when the field is presented.
- The help text to display when the user clicks on the help icon seeking additional information about this field.
- and more

The following sections further discusses concepts and guidelines related to defining fields.

Physical vs. Work Fields

A Field definition record exists for every physical column on a database table that is used by the application. Refer to Understanding Tables for more information.

A Field that does not represent a physical column on a database table is considered as work field and needs to be marked as such. Work fields are typically used as labels as well as describe internal elements used in business rules.

Data Type Information

The type of data captured by a field is defined using the following attributes:

- **Data Type** indicates the basic type of value the field holds. Valid values are **Character, Character Large Object, Date, DateTime, Number, Time, Varchar2** and **XML Type**.
- **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, Switch** and **URI**.
- **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.
- **Scale** is only applicable for number fields. It indicates the number of decimal places supported by the field.
- **Sign** is only applicable for numbers. It indicates if the data may contain positive or negative numbers.

Data Type Inheritance

A Field can be defined to inherit its data type attributes from the definition of another field. The field to inherit the attributes from is referenced as the **Base Field**. When using a base field reference, the data type attributes are defined once, on the base field, and all the fields inheriting their data type information from it are kept consistent with that definition as it may change over time. Any change of a field's data type attributes is automatically reflected on all the fields that reference it as their base field.

Data type information may either be specified directly on the field itself or by referencing a base field. The system does not support a way to partially override the base field's data type information.

It is recommended to leverage data type inheritance whenever possible, especially when the reason for creating a new field is to provide an alternate label for the original field in a certain context. In that situation the new field should reference the original field as its base field and provide the new labels and help text as needed.

Labels and Help Text

For fields that appear on a user interface, the description acts as a label to be used for presentation purposes. As per the product's multi-language support, a field is associated with a label for each of the languages enabled in the system.

A field's help text may be used to provide the user with additional information on how the field is used. When populated for a field that is visible on a user interface that supports display of embedded help, a help icon appears for the field, presenting the help text to the user when clicked.

Overriding Labels and Help Text

The label and help text provided for a product owned field may be overridden by your organization using the corresponding override fields.

The field's label and help text can be overridden for a specific table by populating the corresponding override information as part of the table / field definition. 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. For fixed pages, if the override label at the table / field level takes precedence over the override at the field level.

Alternate Labels

There are contexts where a separate label may be needed for the "date" and "time" parts of a date/time field. Such an example is when the field is used by the operational analytics product. For example, for a "Completion Date/Time" field you may provide the context label "Creation Date" for the **Date** context and "Creation Time" for the **Time** context. These context labels are optional.

In the same way, date fields, time fields and date/time fields may be associated with a **Date and Time Base** context label, which is a label without a reference to the word "Date", "Time" or "Date / Time". For example, for a "Completion Date/ Time" field, the date and time base label may be "Completion". This label that can be used in combination with date or time attributes such as day, month, year, hours, minutes etc.

Overrides labels are available for these additional context labels.

Tables

A Table definition describes the metadata needed by the application to work with and access a database table or view.

The definition includes the following key details:

- The table's description in the application.
- The list of fields defined on the table and any foreign key constraints they establish with other tables.
- The table's prime key and how it is generated.
- Whether the table holds administrative, master or transactional data.
- Whether the table is released with product owned system data rows or not.
- The table's role in the conversion process.
- How should table content be upgraded.
- and more

The following sections further discusses concepts and guidelines related to defining tables.

System Tables

Tables released with rows that are owned by the base product are referred to as system tables. These tables include the **Owner** (OWNER_FLG) field to capture the owner of each row.

Refer to System Data for more information about naming conventions and standards around row record ownership.

Designated Table Roles

Some tables are designed to support the following application roles:

- **Key Tables**
  - A key table holds a copy of a table's prime key only and is only applicable to tables with system generated keys.
  - Tables with a system generated keys reference their Key Table as part of their Table definition record.
  - Records in the key table remain in the system even when the related entity is archived, preventing the same key from being allocated to a new record. Retaining the key allows the system to keep the integrity of any references that may still exist to the archived entity.
- **Language Tables**
  - To support multi-language implementations, all labels and descriptions that are presented to a user are kept in a designated language table where a row exists for each enabled language.
  - The product is released with the English content provided in the various language tables. An additional row is added to these tables for each enabled language along with the translated content for that language. Refer to Support For Different Languages for more information.
  - Tables associated with language sensitive fields reference their corresponding Language Table as part of their Table definition record.
- **Characteristics Tables**
  - Many entities in the system support one or more Characteristics tables, which are used to capture additional fields for the object that are not already supported by the entity's provided attributes.
  - A Characteristics table is designed to capture characteristics applicable to its specific entity only. The entity associated with a Characteristics table is referenced on the Table's definition record.
  - Refer to Setting Up Characteristic Types & Values for more information.
- **Entity Log Tables**
  - Many entities in the system support the ability to capture lifecycle and other important business events on a designated log table.
  - A log table captures the event details in the form of an application message and stores any parameters for the message in a child log parameters table. All log and log parameters tables follow a predefined structure allowing them to be consistently and efficiently implemented.
  - Typically log tables do not participate in the conversion process.

Table Constraints

Foreign key relationships between tables are not enforced at the database level but rather at the application level, providing the application more flexibility around data manipulation and referential integrity.

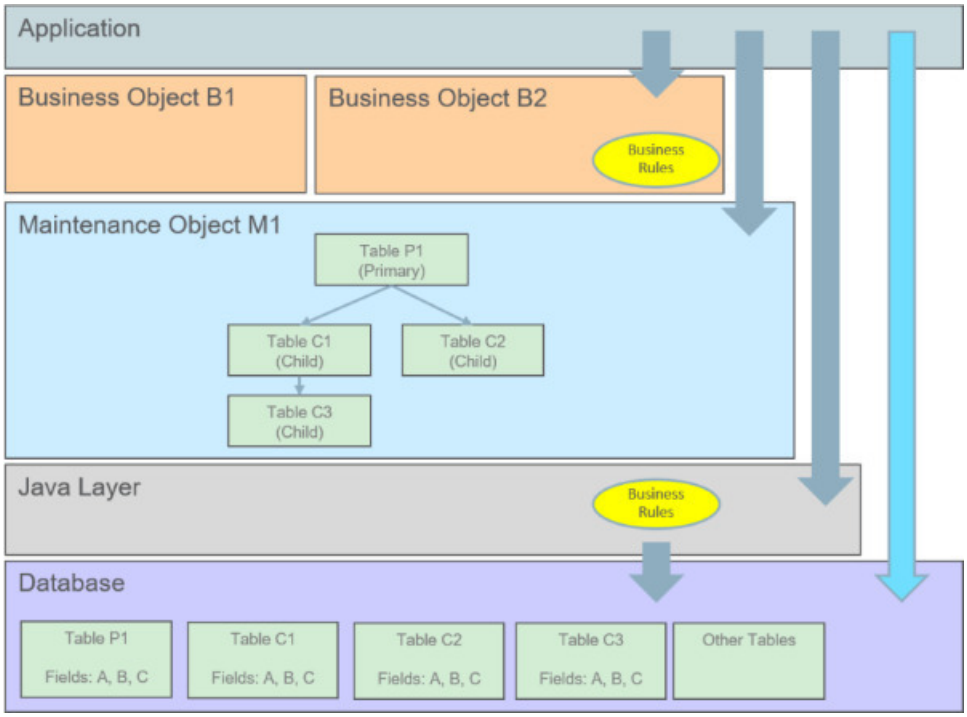
The following types of constraints may be defined on the Table's definition record:

- **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.

The system leverages a table's foreign key constraints information to performs referential integrity validation when rows in this table are deleted. These validations can be disabled by checking off the **Enable Referential Integrity** field on the Table definition record.

Maintenance Objects

A maintenance object defines the configuration of a given “entity” in the system. It includes a primary table and any additional child tables that together capture the physical data for the entity. The application reads, adds, updates and deletes an entity as a single object by invoking its maintenance object service.



The following points briefly outline key aspects of application’s interaction with maintenance objects as illustrated in this diagram:

- Database tables are maintained solely via the application’s Java layer. Direct updates to a table via its Java class is not common and typically reserved to special processes.
- A maintenance object orchestrates the update to an entity’s primary table and its child tables as a single object.
- A business object defines a logical representation of an entity, including all or a subset of the maintenance object’s tables and fields.
- The more common way to maintain an entity is via its maintenance object, either directly or via one of its business objects. Most entities support the use of business objects and as such their maintenance is typically made via their business objects.
- Direct access to database tables is supported for query purposes only.

The following sections further discusses concepts and guidelines related to defining tables.

Maintaining An Entire Object

A maintenance object includes a single primary table and one or more child tables where each child table can further have nested child tables. The table structure of a maintenance object represents the scope of data that should be maintained together as a single object:

- When an object is added, data in added to each corresponding table.
- When an object is updated, data in updated in each corresponding table.
- When an object is read, data in retrieved from all the tables.
- When an object is deleted, data in deleted from all the tables.

Algorithms

Several algorithm plug-in spots are also defined on the maintenance object, allowing for business rules that govern all records for this maintenance object.

Options

The maintenance object includes options that define important information related to the maintenance object that may be accessed for logic throughout the system.

Maintenance Objects vs. Business Objects

A maintenance object defines the physical structure of an entity as stored in the database. Different business objects may be defined on top of a maintenance object to represent different types of such entity. A business object includes only the relevant information applicable to the specific type of entity as well as enforces business rules applicable only to that type.

When a maintenance object supports business object configuration, the system uses the **Determine Business Object** maintenance object algorithm to determine the identifying business object that controls a given entity's data structure and rules.

Refer to The Big Picture of Business Objects for more information.

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.

Characteristic Types

Many maintenance objects in the system support characteristics tables, 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.

While each characteristic is stored in a separate row in the characteristic table, the user interface metaphor used to capture and display them 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 the object. The display and maintenance of each characteristic is determined by the business object's user interface design. For these types of pages, it is recommended that characteristics are defined as flattened elements allowing the user interface to display the characteristic as if it is another field. Refer to The Flattening Nodes and Attributes for more information flattening characteristics on a business object schema.

- 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 various aspects of a characteristic type definition and usage.

Characteristic Value Types

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 "skill", the information you setup on this characteristic type controls the valid values that may be specified by a user when defining another user's skill.

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. 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.
- **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.

  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. 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.

  Also note that the functionality described in the topics for Referencing URIs do not apply to this value given that the browser is responsible for connecting to the URI and does not go via server logic.

  For references to a file, the recommendation is to use the Attachment functionality to link a file 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. To confirm if a given characteristic table supports searching, verify that it includes the field **SRCH_CHAR_VAL**.

> **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**.

Foreign Key References

A Foreign Key Reference defines the necessary information needed to reference an entity in a 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 Characteristic Types 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.
- 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.
- The name of the search page or zone used to look for a valid entity.

The topics in this section describe various aspects of a foreign key reference definition and usage.

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 information.

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 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.

- 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.

NOTE: Not every FK reference provided with the product is configured with a search option. This may be the case if the base delivered pages use a dropdown for this foreign key rather than a search. In addition, base provided FK references that do provide a search may not be configured with both search options. It means that if linking this type of object as a characteristic, the search may not be available if the appropriate search is not configured.

Defining Tables

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

Refer to Tables for more information.

Table - Main

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

Use the **Table Query** portal to search for an existing table definition. Once a table is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 captured as entered and displayed as captured in the database. The assumption is that all date / times reference the time zone as defined on the installation options. Refer to Standard vs. Legal for more information.

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 if using the legacy table / field audit technique. 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.

> **NOTE:** In general, randomly generated system keys are used to attempt to evenly distribute records across a full range of possible IDs. Batch programs that use multiple threads will typically divide the threads using ID ranges and evenly distributed keys will help spread out the work. The reason for inherited keys for child records further extends the performance benefit. When considering partitioning, the recommendation for DBAs is to range partition data based on the primary key so that different batch threads operate on different partitions which reduces contention for hot blocks. Ideally the number of batch threads will be an exact multiple of the number of partitions. Batch programs that insert child data (for example batch Billing creation) also benefit from this design especially when the child tables are partitioned in the same way. The parent is often the driver of the batch process. If this is multi-threaded, then each thread is processing a set of parent records in a given ID range and all child records are being inserted into the same ID range.

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**. Refer to Standard vs. Legal for more information.

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 Fields

This portal is used to view and maintain information about a field. For product owned fields, you may use this portal to override the label of a field on the various pages in the system.

You can access the portal from the**Admin Menu** > **Database** > **Field**. You are brought to a query portal with options for searching for a specific field. Once a field has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to Fields for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Field.** This zone provides information about the selected field.
- **Tables Using Field.** This zone lists tables referencing the field. It is not visible for work fields.

Defining Maintenance Objects

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 Maintenance Objects for more information.

Maintenance Object - Main

Navigate to**Admin** > **Database** > **Maintenance Object** to view information about a maintenance object.

Use the **Maintenance Object Query** portal to search for an existing maintenance object. Once a maintenance object is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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 a transaction and all MO audit algorithms are executed in order of when each record was first added or updated.
Determine BO	Optional	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. 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.
ILM Eligibility	Optional	Algorithms of this type are used for maintenance objects that are enabled for object erasure forInformation Lifecycle Management. They are used to review records that have reached the maximum retention days and evaluate if they are ready to be archived. 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.
Information	Optional	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. 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.
Manage Erasure Schedule	Optional	Algorithms of this type are used for maintenance objects that are enabled for object erasure, which is a measure to protect Data Privacy. They are triggered when certain system events occur for those objects. The algorithms are used to determine if the record

System Event	Optional / Required	Description
		needs to be scheduled for erasure and, if so, create or maintain an entry for the record in the Object Erasure Schedule.
Revision Control	Optional	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.
Transition	Optional	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. 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.
Transition Error	Optional	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. 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. 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.

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 Lookups

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.

Use the **Lookup Query** portal to search for an existing lookup. Once a lookup is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Extendable Lookup Advanced Topics

This section provides some addition technical information about extendable lookup attributes

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 value is known when the attribute is needed (in which case a CLOB mapping is appropriate) or if the functionality expects to determine the lookup value 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.

Capturing a Password

If an extendable lookup includes configuration of a password for some functionality, the system supports automatic encryption of the password value if the schema maps the password to a characteristic using the characteristic type **F1-PWD**. Refer to the business object **F1-FileStorage** for an example of such a configuration.

Defining Characteristic Types

The topics in this section describe the transaction that allows you to define characteristic types.

Refer to Characteristic Types for more information.

Characteristic Type - Main

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

Use the **Characteristic Type Query** portal to search for an existing characteristic type. Once a characteristic type is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.
- **File location value.** Characteristics of this type contain a URI. The URI can point to a file or any web site. Refer to There Are Four Types Of Characteristics for limitations associated with this type of characteristic value.
- **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.

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.

Defining Foreign Key References

The topics in this section describe the transaction that allows you to define foreign key references.

Refer to Foreign Key References for more information.

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

Use the **Foreign Key Reference Query** portal to search for an existing foreign key reference. Once a foreign key reference is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Audit Trails

The topics in this section describe one way of auditing changes in the system. Note that this technique has limitations and may not be the best option for all situations.

- This functionality is configured at the table and field level. Many base maintenance objects use an XML or CLOB field to capture one or more elements using XML format, configured using a BO schema. This auditing technique is not able to capture changes to individual elements. It can only capture overall changes to the single field. The BO Audit plug-in is a better option for auditing changes to individual elements with the CLOB / XML field.
- The base table provided for supporting audits limits the field size of the before and after values to 254 bytes.
- Auditing is captured for each field. In some cases it is prefered to capture a before and after image for several fields at once. For example, if an address is changed, it’s nore user friendly to capture the before and after for the full address rather than the individual address components. The BO Audit plug-in allows for capturing a single audit record for multiple elements rather than granular changes for each element.

The subsequent topics highlight how to enable auditing for fields, and describe the 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. If any of the changed fields are marked for audit, the audit program inserts audit rows into the audit table.

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 the base audit program and base audit tables 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 base audit insert program 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 base delivered 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.). An appropriate java table name must be defined and an *_impl.java program for that java table name must be defined.
- The name of the new table must be referenced on the various tables whose changes should be logged in the new table.

NOTE: 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 the base delivered audit program 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, 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** . The product provides two programs. **com.splwg.base.domain.common.audit.DefaultTableAuditor** is a standard table audit program. In addition **com.splwg.base.domain.common.audit.ModifiedTableAuditor**is supplied as an alternative. This program does not audit inserts or deletes of empty string data.

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: 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 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**.
- **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.

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.

Note that use of the bundling add BO also benefits records that have optional foreign keys or foreign keys that are part of a child table. That way the person creating the bundle does not have to worry about the sequence of the records. For example, an FK Reference may optionally reference a zone for searching. If a new FK Reference and its search zone are bundled together to copy to another region, the bundling add BO for FK Reference (which doesn’t include the Zone) ensures that the FK reference could be added before the zone without getting any validation errors.

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:

Configuration Task	Scope of Task
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	Only needed if there are objects with recursive or optional 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, optional foreign key references or child tables that include foreign key references, 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 child tables and all elements that are not required.
4. Save the 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.

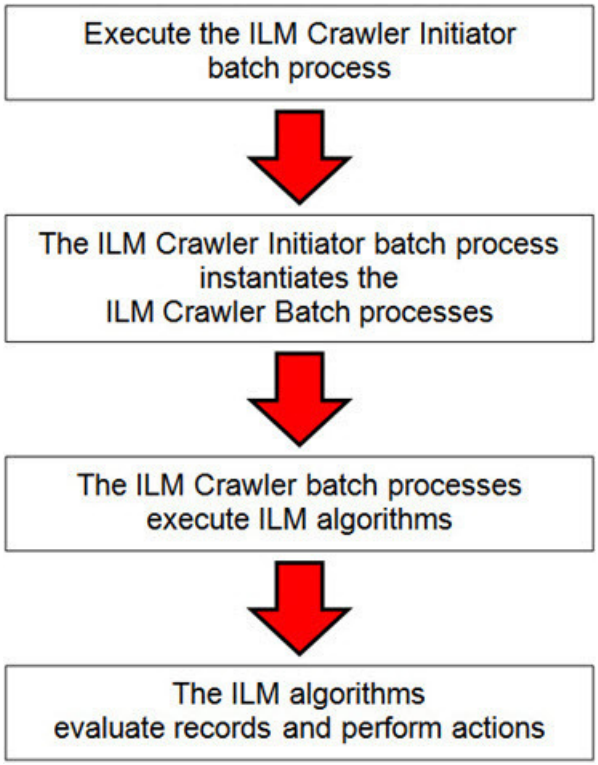
Batch Processes

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 embedded 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 chooses 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.

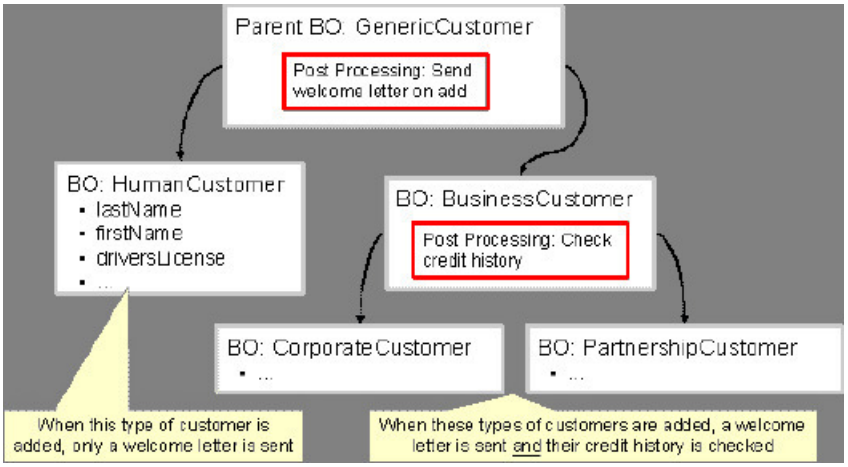
For all maintenance objects, the base product may provide additional BOs that are not meant to be “identifying” BOs, but instead are provided to support functionality to interact with the MO using the BO as a tool as described in Invoking a BO.

- 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.
- A “bundling add” BO may be supplied. Refer to Recursive Key References for more information as to why this type of BO may be supplied.

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.

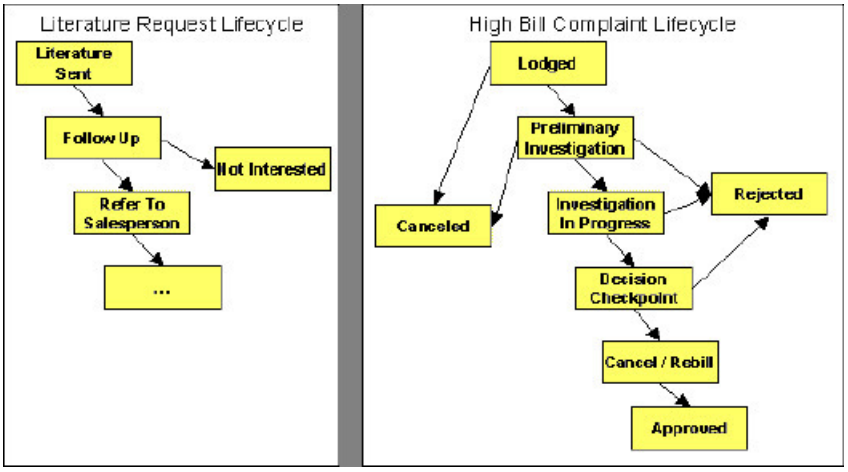
NOTE: User interface maps are inherited. When determining if the business object has a UI map to use for UI rendering, the system looks for display and maintenance maps linked to the BO as options. If the identifying BO does not have maps defined, the system follows “up the chain” of inheritance until 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.

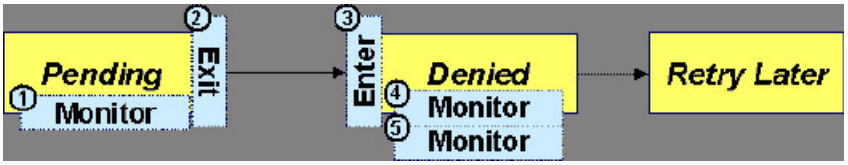
The system executes all state-specific algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

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 Error**Maintenance 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 (referred to as "deferred"), 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 selects records whose current state references this particular batch control as their monitor process. Deferred monitor process is only needed when an object has different use cases for monitoring the same type of records with different schedules. In this case only one periodic monitor batch should be configured. Other monitors should be configured to restrict by batch control so that there is no overlap in processing.

NOTE: MO option configuration. The maintenance object includes options to indicate the batch controls delivered for periodic and deferred monitor batch controls.

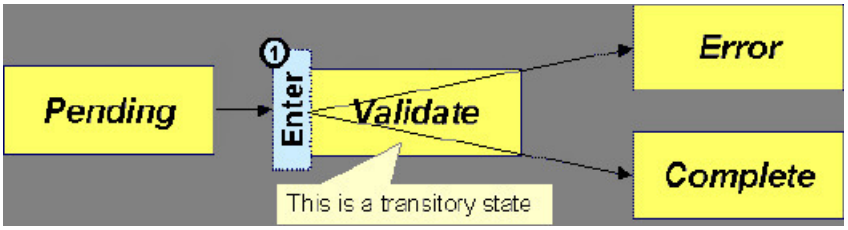
Your business rules will dictate the execution frequency of each monitoring process and the order in which they should be scheduled. Refer to Monitor Background Processes in the background process chapter for more information about the parameters supported for this type of batch process.

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 captures 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.

Opening UI Maps Before Entering A State

You can define a UI Map to capture additional elements before a business entity can enter a given state. You do this by configuring the **State Transition UI Map** state-specific option on the appropriate state on the business object.

There may be circumstances in which the status pre-processing logic can determine the value of the elements, in which case it is not necessary to invoke the map. The system provides a data area (**F1-StateTransitionCommon**) which contains a boolean element to indicate if the state transition map can be skipped. This data area can be included in your pre-processing script to allow the element to be set according to your business logic. The base maintenance script (**F1-MainProc**) references this element to determine whether to open the state transition map.

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.	BO Status Exit algorithms for the “old” status executed.
	Status updated to the new value.
	BO Status Enter algorithms for the “new” status executed.
	If no error — MO Transition algorithms are executed. **FASTPATH:** Refer to State Transitions are Audited for more information.
	If error and there are “save points” the MO Transition Error algorithms are executed. **FASTPATH:** Refer to Keeping An Entity In Its Last Successful State for more information. Otherwise, the error is reported.
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**

Use the **Business Object Query** portal to search for an existing business object. Once a business object is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

The options grid allows you to configure the business 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 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.

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 that each maintenance object has an underlying service and it's possible that some of the child tables in the maintenance object are not included in the service definition. If that is the case, a message is issued and the generated schema would need to be adjusted to remove that child table.

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	Algorithms of this type may be used to audit certain changes made to business object instances. 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. It is the responsibility of the algorithms to record corresponding audit information.

System Event	Optional / Required	Description
		The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.
Information	Optional	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. 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.
Post-Processing	Optional	Algorithms of this type may be used to perform additional business logic after a business object instance has been processed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.
Pre-Processing	Optional	Algorithms of this type further populates a request to maintain a business object instance right before it is processed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.
Validation	Optional	Algorithms of this type may be used to validate a business object instance when added, updated or deleted. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.

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** 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.
Exit	Optional	Algorithms of this type apply business rules when a business object instance exits 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.
Monitor	Optional	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. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object Inheritance for more information.

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 embedded 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 plug-in 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.

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 speical 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**.

Use the **Data Area Query** portal to search for an existing data area. Once a data area is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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.

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.

Application Service is the application service that is used to provide security for the service. If the service is related to a maintenance object, the access modes for the application service should be the standard **Add**, **Change**, **Delete** and **Inquire**. For other services, the application service should have the **Execute** access mode.

Service Type indicates whether the service is a **Java Based Service** or a **Java (Converted) Service**. Note that services generated to support a portal in the system will not have a service type populated.

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.

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**

Use the **Business Service Query** portal to search for an existing business service. Once a business service is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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 NameFWLZDEXP**.
- 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.
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-AddWarning	This business service may be used to log a warning by providing the message details. The system gathers all the warning and displays them to the user all at once.

Business Service Name	Description
F1-CheckOnlineConnection	Check if Connection is Online or Batch.
F1-CheckWarning	Used to check if any warnings have been accumulated. If warnings have been accumulated, any logic that cannot be rolled back (when the warnings are shown) should not be performed.
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. Refer to Real Time Messages for more information.
F1-OutmsgMediator	Alternative to **F1-OutmsgDispatcher** and may be a better option if the sender does not require an outbound message record to be instantiated. Refer to Real Time Messages for more information.
F1-RethrowError	Issues an application error using the input message category / number / parameters.
F1-RetrieveMODescription	Retrieves the description for a maintenance object.
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-DetermineIfUserCanApproveTD	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**

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**

Use the **UI Map Query** portal to search for an existing UI Map. Once a map is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 XHTML Document**, **Complete HTML Document**, **XHTML Fragment** or an **HTML Fragment**. The default value is **Complete XHTML Document**.

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.

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

NOTE: This topic uses the term "field" to refer to both the generic concept of displaying and capturing data in a 'field' as well as referring to the meta-data object supplied in the product to define Fields. When referring to the latter, the term "MD Field" (meta-data Field) is used and a hyperlink to the Field documentation is provided.

Contents

Bind XML to HTML
Build a Dropdown List
Format Input and Output Fields
Display Labels
Enable UI Map Help
Search Using a Pop-Up Explorer Zone
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
Detecting Unsaved Changes
Hiding Portal Tabs

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.

Linking a Field

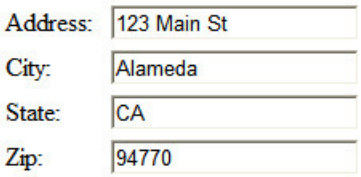
Syntax	Values	Description
oraField=" "	Field element XPath	This attribute is used to link an HTML element directly with an XML element, where the XML element is defined within the UI Map's XML schema. The attribute can be used with any *rendering* HTML element, such as: , , and <input>.

- 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>
```

Rendered HTML



- 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 rendered:

123 Main St
Alameda, CA 94770

Linking a List

This 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 element is to trigger the framework to replicate the table's HTML for each occurrence of the list.

Syntax	Values	Description
oraList=" "	List element XPath	This 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 element is to trigger the

Syntax	Values	Description
		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>
        <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 rendered:

Pay Date	Amount
01-01-2008	$44.28
02-01-2008	$32.87
03-01-2008	$21.76

Build a Dropdown List

The following attributes are provided to build an HTML ‘select’ element, also called a dropdown, based on various sources.

Source	Syntax	Values	Examples
Lookup	oraSelect="lookup: ;"	Lookup field	... <td>House Type:</td> <td> <select oraField="houseType" oraSelect="lookup:HOUSE_TYPE;"> </select>

Source	Syntax	Values	Examples
			</td>
Extendable Lookup	oraSelect="lookupBO: ;"	BO code	... <td>UI Device Display Type:</ td> <td> <select oraField="uiDeviceType" oraSelect="lookupBO: F1- DeviceDisplayTypes;"></ select> </td>
Characteristic Type (pre-defined)	oraSelect="charType: ;"	Characteristic Type code	... <td>Usage:</td> <td> <select oraField="statusReasonUsage" oraSelect="charType:F1- SRUSG;"></select> </td> ...
Control Table	oraSelect="table: ;" **NOTE:** This attribute only works with tables that follow the standard control table structure where there is a related language table that includes that includes the column **DESCR** as its description column. Use the Application Viewer data dictionary to identify tables that qualify for this functionality. **WARNING:** The oraSelect function will only work if less than 500 values are displayed.	Table name	... <td>Currency: </td> <td> <select oraField="currency" oraSelect="table:CI_CURRENCY_CD;"> </select> </td> ...
Page Service	oraSelect="service: ;"	Page Service name	... <td>Country:</td> <td> <select oraField="country" oraSelect="service:CIPTCNTW;"> </select> </td> ...
Embedded List	Used to build a select dropdown based on a list within the map's XML. oraSelect="valuePath: ;descPath: "	After the valuePath, indicate the XPath of the element that holds the values. After the descPath, indicate the XPath of the element that holds the descriptions.	<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> <xml> <root> <target>10</target> <list> <value>10</value> <desc>Ten</desc>

Source	Syntax	Values	Examples
			</list> <list> <value>20</value> <desc>Twenty</desc> </list> <list> <value>40</value> <desc>Forty</desc> </list> </root> </xml> </html>
Service Script	oraSelect="ss: ;"	Service Script code	See below for additional syntax needed when using this function.
Business Service	oraSelect="bs: ;"	Business Service code	See below for additional syntax needed when using this function.

When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.

Syntax	Values	Description
oraSelectIn=" ;"	serviceXPath:element;	Used to pass the value of another element into the service (mapping to the service's XPath).
	serviceXPath:'Literal';	Used to pass a constant or literal to the service (mapping to the service's XPath).
oraSelectOut="valuePath: ; descPath: "	See examples below	Used to indicate which element in the service's output holds the values and which one holds the descriptions.

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>
```

```
...
```

Format Input and Output Fields

The following attributes are designed to apply data type formatting for input and output fields.

Automatic Formatting

Syntax

oraSchemaDataTypes="false"

This attribute is used to trigger automatic formatting in the rendered HTML document. The automated formatting will occur according to the data type 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>
```

HTML rendered.

dataType	result type	input result		display-only result
date	raw	2007-11-02		2007-11-02
	rendered	11-02-2007		11-02-2007
dateTime	raw	2007-11-02-23.45.00		2007-11-02-23.45.00
	rendered	11-02-2007	11:45PM	11-02-2007 11:45PM
fkRef	raw	SYSUSER		SYSUSER
	rendered	SYSUSER		SYSUSER
lookup	raw	A		A
	rendered	Add		Add
money	raw	1000000		1000000
	rendered	$1,000,000.00		$1,000,000.00
number	raw	5661976.11548		5661976.11548
	rendered	5,661,976.11548		5,661,976.11548
time	raw	23.45.00		23.45.00
	rendered	11:45PM		11:45PM

Date Formatting

This function is used to display a date according to the user's display profile. For input fields, the setting formats the data when the user tabs out of the field.

Syntax

oraType="date"

```
<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>
```

HTML rendered.

Date: 12-28-2008

Date: 12-28-2008

Time Formatting

This function is used to display a time according to the user's display profile. For input fields, the setting formats the data when the user tabs out of the field.

Syntax

oraType="time"

```
<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>
```

HTML rendered.

Time: 12:28AM

Time: 12:28AM

Date and Time Formatting

This function is used to display a timestamp according to the user's display profile. If this function is used for an input element, it is broken into two pieces, one for date and one for time. Optionally, the time portion of the date time element can be suppressed using the attribute value 'time:suppress'.

Syntax

oraType="dateTime; 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>
```

HTML rendered.

Date time: 11-01-2009 12:28AM

Date only: 11-01-2009

Date time: 11-01-2009 | 12:28AM

Date only: 11-01-2009

Date / Time Formatting with Standard Time

This 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. For input elements with this setting, 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.

Syntax

oraType="dateTime; stdTime:true;"

```
<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>
```

HTML rendered.

> **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: 11-01-2009 | 01:28AM PDT

HTML rendered for the following day.

Date time: 11-02-2009 12:28AM

Date time: 11-02-2009 | 12:28AM

Date and Time Formatting with Time Zone Reference

Syntax	Valid Values	Description
oraType="dateTime; stdTimeRef: ;"	Reference an XPath after the colon.	This function is used to render a date / time element according to the daylight savings time schedule of a time zone whose XPath is referenced. 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.
oraType="dateTime; displayRef: ;"	Reference an XPath after the colon.	This function is similar to the stdTimeRef function, except that this function 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.

For input elements, 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>
```

HTML rendered.

NOTE: The time zone label is always displayed for a referenced time zone.

Date time: 11-01-2009 01:28AM EDT

Date time: 11-01-2009 | 01:28AM EDT

HTML rendered for the following day.

Date time: 11-02-2009 12:28AM EST

Date time: 11-02-2009 | 12:28AM EST

Duration Formatting

Syntax

oraType="duration"

This function is used to display time duration. For an input element, the value entered by the user is translated from minutes to hour and minutes as appropriate. For example, an entered value of '90', is converted to '00:01:30' when tabbing 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>
```

HTML rendered.

Duration: 00:01:30

Duration: 00:01:30

Day in Month Formatting

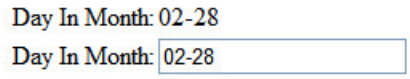
Syntax

oraType="dayInMonth"

This function is used to display a concatenated day and month.

```
<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>
```

HTML rendered.



Month In Year Formatting

Syntax
oraType="monthInYear"

This function is used to display a concatenated month and year.

```
<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>
```

HTML rendered.

Monetary Formatting

This function is used to display a number as a monetary amount. See the table for configuration options with respect to the currency. For input elements, an error is issued if a non-numeric value is entered.

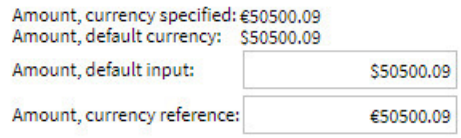
Syntax	Description
oraType="money: "	Directly specify a currency code after the colon.
oraType="money;currencyRef: "	Reference an XPath (after the colon) for an element that references a currency code.
oraType="money"	If no currency or currency reference is specified, the installation currency is used.

NOTE: You must specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header for right alignment. 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>
<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>
```

HTML rendered



Number Formatting

This function is used to display a number or validate an input value. For input elements, the system will return an error if a non-numeric value is entered.

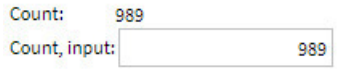
Syntax
oraType="number"

NOTE: You must specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header for right alignment. 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>
```

HTML rendered.



FK Reference Formatting

By default, when an element with an **fkRef** oraType 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.

Note that you can enable the foreign key hyperlink separately as well, refer to Embed Framework Navigation 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.

Syntax

oraType="fkRef:true|false; info:true|false; context:true|false; navigation:true|false; search:true|false"

- **fkRef**. A value of 'true' enables all of the following foreign key reference processing. Use a value of 'false' to disable automatic FK Reference processing.
- **info**. A value of 'true' renders an information string on the UI map, if applicable.
- **context**. A value of 'true' renders a context menu to appear before the foreign key reference element, if applicable.
- **navigation**. A value of 'true' causes the information string to be rendered as a hyperlink, if applicable. Clicking the hyperlink navigates to the appropriate page.
- **search**. A value of 'true' displays a search icon that launches the search zone if applicable.

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>
  <bo fkRef="F1-BOMO"/>
</schema>
```

- UI Map HTML:

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
<td>Business Object</td>
<td><span oraField="bo" oraType="fkRef:true; info:true; context:true; navigation:true;"></
span></td>
    </tr>
</table>
</body>
<xml>
<root>
<bo>F1-COUNTRY</bo>
</root>
```

```
</xml>
</html>
```

- HTML rendered.

Lookup Formatting

This function is used to display the description of a lookup value.

Syntax	Valid Values
oraType="lookup: "	Lookup field name after the colon.

```
<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>
<xml>
<root>
    <status>PD</status>
</root>
</xml>
</html>
```

HTML rendered.

Status:	Pending

Extendable Lookup Formatting

This function is used to display the description of an extendable lookup value.

Syntax	Valid Values
oraType="lookupBO: "	Business Object code name after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
        <td>Value:</td>
        <td><span oraField="status" oraType="lookupBO:F1-DeviceDisplayTypes"></span></td>
    </tr>
</table>
</body>
<xml>
<root>
    <status>oraTablet</status>
</root>
</xml>
</html>
```

HTML rendered.

Value:	Tablet size

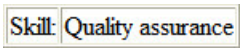
Characteristic Type Formatting

This function is used to display the description of a predefined characteristic value.

Syntax	Valid Values
oraType="charType: "	Characteristic Type code after the colon.

```
<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>
```

HTML rendered.



Control Table Formatting

This function is used to display the description of a control table that has an associated language table.

Syntax	Valid Values
oraType="table: "	Table code after the colon.

```
<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>
```

HTML rendered.

Add / Remove Grid Formatting

Syntax	Description
oraType="addGridRow"	The addGridRow function is used to build "insert row" dialog into the UI map.

Syntax	Description
	• An 'add' image is displayed. • Clicking the image inserts a new row in the grid. • If the list is empty, by default, an empty grid row is automatically be added. This means that the user will always see at least one grid row when this attribute is used.
oraType="deleteGridRow"	The deleteGridRow function is used to build "delete row" dialog into the UI map. • A 'delete' image is displayed. • Clicking the image removes 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.

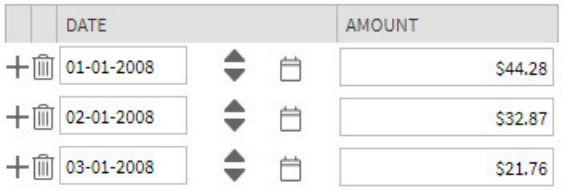
WARNING: These attributes are designed to work with the business object action of 'replace' rather than 'update'. Therefore, if the map contains a grid, the business object action of 'replace' must be used to update the business object. Refer to BO Replace Action for more information.

Example:

```
<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>
        </tr>
        <tr/>
    </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>
```

HTML rendered.



Unformatted Elements

This function is used to display the contents of an element that contains 'raw' data as defined for the schema element being rendered.

Syntax

oraType="raw"

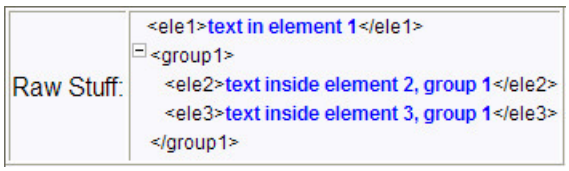
- UI Map schema:

```
<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>
```

HTML rendered.



String Formatting

This function is used to display the contents of an element, as XML pretty-print, when the element contains escaped XML.

Syntax

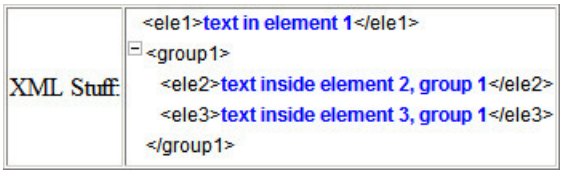
oraType="xmlString"

> **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.

Example:

```
<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>
```

HTML rendered.



HTML rendered without oraType=”xmlString”

XML Stuff:	<ele1>text in element 1</ele1><group1><ele2>text inside element 2, group 1</ele2><ele3>text inside element 3, group 1</ele3></group1>

HTML Formatting

This 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.

Syntax
oraType="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>
```

HTML rendered.

Info : **text in bold**

HTML rendered without oraType="html"

Info : text in bold

Display Labels

Derive Label from an Element

This attribute is used to obtain a language sensitive label for a , <td>, or <input> HTML element.

Syntax	Valid Values
oraLabel=" "	XPath of an element in the UI map schema. The element must reference either the **mapField=**, **mdField=**, or **label=** attribute.

NOTE: You can also define a field directly in your HTML for label definition, refer to Deriving Label from a Field for more information.

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:

Deriving Label from a Field

This 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.

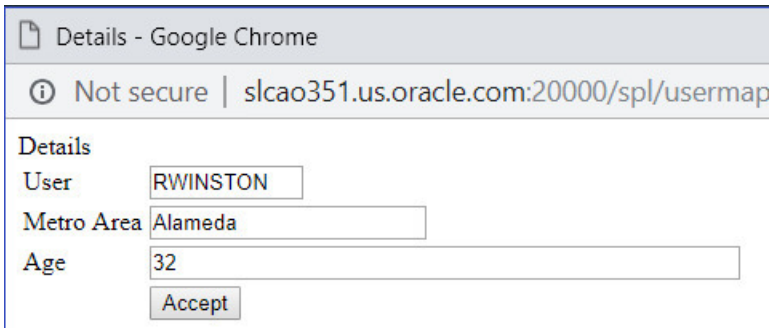
Syntax	Valid Values
oraMdLabel=" "	MD Field code.

NOTE: You can also define labels derived from the map's schema definition, refer to Derive Label from an Element for more information.

- HTML:

```
<html>
<head><title oraMdLabel="F1_DETAILS_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:



Enable UI Map Help

The Display Labels section describes ways to derive the label for an element using an underlying MD Field. In addition, if the same MD field contains help text, the system will automatically generate a tool tip adjacent to the element label. Clicking the tool tip allows the user to view the help text.

It is possible to change the rendering of the tool tip. Refer to Custom Look And Feel Options for more information

Search Using a Pop-Up Explorer Zone

Search Option

This attribute is used to enable search zone functionality for input HTML elements.

Syntax	Valid Values
oraSearch=" "	Zone code.

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 FK reference.

- Example of defining the search in the HTML. UI Map's Schema:

```
<schema>
    <uiMap/>
</schema>
```

UI Map's HTML

```
<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>
<xml>
<root>
    <uiMap/>
</root>
</xml>
</html>
```

- Example of defining the search in the schema. UI Map's Schema:

```
<schema>
    <uiMap search="F1-UISRCH"/>
</schema>
```

UI Map's HTML

```
<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>
```

- Example where the FK reference defines the search zone. UI Map's Schema:

```
<schema>
    <uiMap fkRef="F1-UISRC"/>
```

```
</schema>
```

UI Map's HTML

```
<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>
```

In all cases, the HTML rendered is the same.

Initializing Search Fields

This optional attribute is used to initialize search zone filters. Multiple filters may be initialized. This attribute can only be used in conjunction with the oraSearch attribute.

Syntax	Valid Values	Field Value Options	Comments
oraSearchField=" "	One or more pairs of field name and field value separated by colon. Each pair is separated by a semi-colon. **oraSearchField="fieldName:fieldValue: ..."** 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 user filter or hidden filter parameter.	No value	If you do not specify a field value, then the value of the input element containing the oraSearchField attribute will be used.
		XPath	Indicate the XPath to the schema element that contains the value to use.
		'literal'	Indicate a literal value to supply.

NOTE: If you do not specify an oraSearchField attribute and the schema element has a search enabled fkRef specified, the framework automatically builds an oraSearchField where the field name is equal to the FK reference's key (MD) fields.

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>
<xml>
<root>
    <bo/>
    <uiMap/>
</root>
</xml>
</html>
```

Mapping Returned Search Fields

This optional attribute is used to direct values returned by the search zone. Multiple fields may be specified. This attribute can only be used in conjunction with the oraSearch attribute.

Syntax	Valid Values	Field Value Options	Comments
oraSearchOut=" "	One or more pairs of field name and field value separated by colon. Each pair is separated by a semi-colon.	No value	If you do not specify a field value, then the input element containing the oraSearchField attribute receives the returned value.
	oraSearchOut="field name:xpath target; ..." 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.	XPath	Indicate the XPath to the schema element that should receive the returned value.

NOTE: 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>
<xml>
<root>
    <bo/>
    <mo/>
</root>
</xml>
</html>
```

Display Errors

Display Error Variables

One of the following error variables may be displayed.

Syntax
oraErrorVar="ERRMSG-TEXT"
oraErrorVar="ERRMSG-LONG"
oraErrorVar="ERRMSG-CATEGORY"
oraErrorVar="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>
...
```

HTML rendered

Highlight a Field in Error

NOTE: For more information on throwing an error, refer to the Terminate Statement in the Edit Data Syntax.

Syntax	Comments
oraError="automate:true|false; prefix: "	Specifying **automate:true** automatically enables highlighting of the element in error when issuing an error. Note that **true** is the default and doesn't need to be specified. Specify **automate:false** to turn off field highlighting.
	The system uses a match on the element name referenced in the error to the element names in the UI map. If the elements in the schema are within an XPath that may not match what is referenced by the error, use **prefix:XPath** to specify that.

NOTE: A pair of stylesheet references, cisEnabled and cisDisabled, must be specified for reference of the oraError style. The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the oraError style.

The following HTML example shows that the elements in the map are defined within a group called **boGroup**. The element name returned by the error will not include this group so in order for the field highlighting to work properly the **prefix:** attribute must indicate the group name.

```
<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>

  <tr>
    <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>
```

HTML rendered, where the error element thrown is equal to 'lastName':

Overriding the Error Element Name

In the rare occasion where the element name returned by the error doesn't match the element name in the map, you can add an explicit attribute to indicate the error element name.

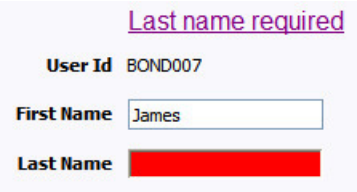
Syntax	Valid Values	Comments
oraErrorElement=	"element name"	The element name referenced here must exactly match the name of the error element assigned when the error was thrown. More than one HTML field can be referenced by the same error element name.

NOTE: A pair of stylesheet references, cisEnabled and cisDisabled, must be specified for reference of the oraError style. The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the oraError style.

This illustrates a scenario where the element name associated with the error differs from the element name on the map.

```
<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>
        <span 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="familyName" class="normal"  oraErrorElement="lastName"/>
    </td>
  </tr>
</table>
</body>

<xml>
<root>
    <userId>BOND007</userId>
    <firstName>James</firstName>
    <familyName></familyName>
</root>
</xml>
</html>
```

HTML rendered.



Display Error Pop-Up

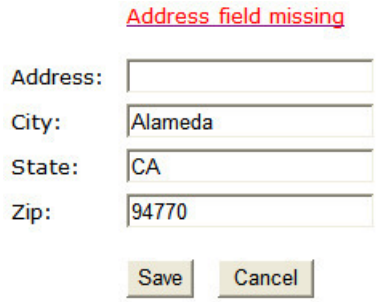
When the error text is displayed, this function may be used to pop-up the standard error dialog (which displays more information) when a user clicks the error message.

Syntax

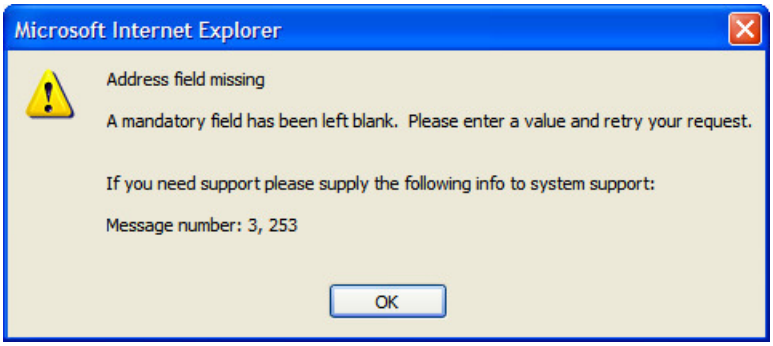
oraShowErrorAlert(); return false;

```
<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-SaveCancelButtons"/>
              </td>
     </tr>
</table>
</body>
<xml>
     <root>
              <address>123 Main St</address>
              <city>Alameda</city>
              <state>CA</state>
              <zip>94770</zip>
     </root>
</xml>
</html>
```

HTML rendered.



Standard error pop-up dialog launched via click on error message:



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>

Element Change Event

Syntax	Valid Values
oraChange=" "	A JavaScript function.

At the time of UI Map load, if there is an event handler 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>
```

Page Load Event

Syntax	Valid Values
oraLoad=" "	A JavaScript function.

> **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>
```

- HTML rendered

User Id:	SPLAXT

After Page Load Event

Syntax	Valid Values
oraAfterLoad=" "	A JavaScript function.

In the following example the oraGetValueFromScript function is executed after 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_externalSystem');">
      </select>
    </span>

</div>
```

Hide Elements

Hide Using a Function

The system provides multiple techniques for hiding an HTML element.

Syntax	Valid Values	Comments
oraDisplayNone(item);	(item, 'XPath', 'value', 'operator')	Used to hide an element based on the value of another element (referenced using its XPath). Enter a value of ' ' to interrogate a blank value. By default the operator is **'='**. This may be set instead to another operator such as **'!='**, **'>'**, or **'<'**.

Syntax	Valid Values	Comments
	(item, function name,**true** | **false**)	Used to indicate a JavaScript function, which must return a Boolean.
oraIsSuppress="true"		Use this option for fields that need to be hidden always and not dependent on something else. Set this attribute on the HTML element (div element).

- Example where the User Id label is hidden when no User Id value exists.

```
<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>
```

- Example where the Save button is hidden when the user does not have security access to the action of change ('C') for the application service '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>
```

Check User's Security Access

The system provides two functions to check a user's security access to a given application service and access mode. These are commonly used for hiding elements.

Syntax	Parameters
oraHasSecurity( )	'Application Service code'
	'Access Mode'
oraHasSecurityPath('x','y')	'Application Service XPath'
	'Access Mode XPath'

See the previous section for an example of the **oraHasSecurity** function. The following shows an example where the status button is 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

The system provides functions for invoking a business object, business service or service script.

Invoke BO Function

This function is used to perform a BO interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oraInvokeBO( )	'BO Name'	
	'XPath' or **null**	Identifies a group element via XPath. If you specify the word **null**, then the entire embedded XML object will be passed.
	'action'	Indicate the action to use. Valid values are **add**, **delete**, **read**, **update**, **replace**, **fastAdd** and **fastUpdate**. **FASTPATH:** Refer to BO Actions for more information about the various BO actions.

- Example with the statement invoked in a JavaScript function.

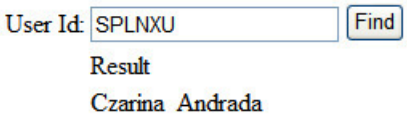
```
function invokeBO {
     if (!oraInvokeBO('F1-User','xmlGroup','read')) {
          oraShowErrorAlert();
          return;
}
}
```

- Example with the statement invoked within onClick.

```
<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('F1-
User','xmlGroup','read');"/>
                </td>
            </tr>
            <tr>
                <td/>
                <td>Result</td>
            </tr>
            <tr>
                <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>
```

HTML rendered.



Invoke BS Function

This function is used to perform a business service interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oraInvokeBS( )	'BS Name'	
	'XPath' or **null**	Identifies a group element via XPath. If you specify the word **null**, then the entire embedded XML object will be passed.

- Example with the statement coded within a JavaScript function.

```
function invokeBS {
     if (!oraInvokeBS('F1-UserSearch','xmlGroup')) {
          oraShowErrorAlert();
          return;
  }
}
```

- Example with the statement invoked via onClick.

```
<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('F1-
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>
```

HTML rendered.

Invoke SS Function

This function is used to perform a service script interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oraInvokeSS( )	'Service Script Name'	
	'XPath' or **null**	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',`

Syntax	Parameters	Comments
		`null, null, [$SEQUENCEID])`

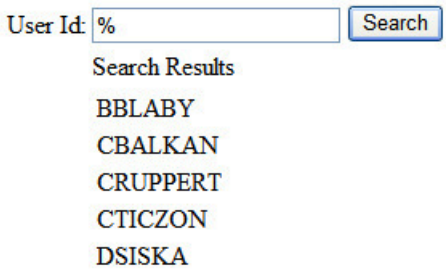
- Example with the statement invoked within a JavaScript function:

```
function invokeSS {
     if (!oraInvokeSS('F1-GetUsers','xmlGroup')) {
          oraShowErrorAlert();
          return;
}
}
```

- Example with the statement invoked within onClick.

```
<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('F1-
GetUsers','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>
```

HTML rendered.



Refresh a Rendered Map or Portal Page

Refresh Map

This function is used to 'Refresh' only the map zone issuing the command.

Syntax

oraRefreshMap;

```
...
    <tr>
        <td/>
        <td><input type="button" onclick="oraRefreshMap();" value="Refresh"/></td>
    </tr>
...
```

Refresh Page

This function is used to refresh all zones in the portal.

Syntax

oraRefreshPage;

```
...
    <tr>
        <td/>
        <td><input type="button" onclick="oraRefreshPage();" value="Refresh"/></td>
    </tr>
...
```

Embed Framework Navigation

Navigate using Navigation Option

This function is used to navigate to another page using the information defined on a navigation option.

Syntax	Parameters
oraNavigate( );	'Navigation Option code'
	'Key XPath'

> **WARNING:** This function is only intended for a UI map defined within a portal zone. It should not be used within a UI map launched by a BPA script.

The following example exhibits two possible uses of this function: as a URL and as a button. Note that the UI Map schema must contain a fkRef attribute. Refer to FK Reference Formatting 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" oraType="fkRef:CI_USER"></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>
```

HTML rendered.

Launch BPA Script

Launch BPA Script

Syntax	Parameters	Comments
oraRunScript( );	'BPA script code'.	
	'XPath Element'	One or more element values may be passed into the BPA where it may be referenced as temporary variables.

WARNING: This 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 rendered.

123 Main St
Alameda, CA 94770 edit

Launch BPA Script With Values

This function is used to launch a BPA, providing name/value pairs to push into temporary storage. Multiple values can be passed. The BPA script can then reference the temporary variables by name.

Syntax	Parameters	Comments
oraRunScriptWithValues( );	'BPA script code'.	
	'XPath Element Name':value	One or more pairs of element names and values.

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

This 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.

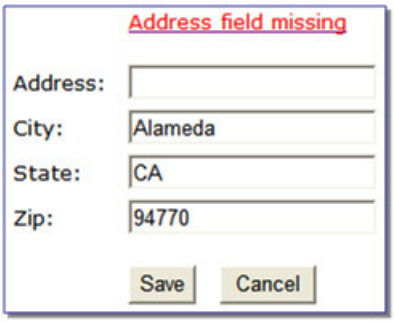
Syntax	Parameters	Comments
oraSubmitMap( );	'Return Value'	
	Boolean value	Indicates if the updated XML should be returned. Default is **true**.

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:



Include a Map Fragment

This 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.

Syntax	Parameters	Comments
<oraInclude map=' ' prefixPath=' '/>	map='UI Map Code'	
	prefixPath='Xpath'	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.

Syntax	Parameters	Comments
		NOTE: This 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

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.

Syntax
<action>ADD</action>
<action> </action>

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>
<xml>
<root>
    <action>ADD</action>
    <boGroup>
        <key/>
        <description/>
    </boGroup>
</root>
</xml>
```

```
</html>
```

HTML rendered.

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 ;"

This 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** **stacked** **cluster** **line** **area** **combo**	Defines the type of chart to display. Required
showLegend	**true** **false**	Defines if the chart should have a legend displayed. Optional (default is **true**)
legendPosition	**left** **right** **bottom** **top**	Defines where the legend should appear. Optional (default is **right**) Setting position to **left** or **right** will automatically render it vertically. Setting position to **top** or **bottom** will automatically render it horizontally.
legendBorder	**true** **false**	Defines if the legend should display with a border around it. Optional (default is **false**)
depth	**true** **false**	A value of **true** indicates a 3 dimensional depth for the chart. Optional (default **false**, which is a 2 dimensional chart)
animate	**true** **false**	A value of **true** indicates that the graph should animate when displayed. Optional (default is **true**). When using large data sets, consider disabling animation.
dataCursor	**on** **off**	A value of **on** enables hovering anywhere in the graph. Optional (default is **off**). It is not applicable to pie charts.

Parameter	Values	Description
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
xaxis:	XPath value	Defines the XPath to the element in the XML list that contains the x-axis data. Required for Stacked, Cluster, Line, Area and Combo charts.
xaxisFormat:	**date** **dateTime** **time** **localDate** **string**	Defines the x-axis data format. If it is **date**, **dateTime** or **time** then the value is presented in the format as defined on the user's display profile. In case of **localDate** or **string** the data is displayed as is with no special formatting. Optional (Default is **date**).
label:	Text value	Defines the label for the amount being charted.

Parameter	Values	Description
		Either this setting or **labelPath:** must be defined.
labelPath:	XPath value	Defines the XPath to the element that provides the label for the amount being charted. Either this setting or **label:** must be defined.
currency:	A valid Currency code	Defines the currency code for the amount being charted. Optional.
currencyPath:	XPath value	Defines the XPath to the element that provides the currency code for the amount being charted. Optional.
hoverText:	Text value	Defines the hover text for the chart element. Optional (A default hover text is always available.) Ignored if **hoverTextPath:** is defined. The following substitution variables are available. • **$label** This will be replaced with the label text as determined by the **label:** or **labelPath:** setting. • **$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. • **$newline** This will be force a line break. If the hover text defined contains any of the above values, they will be replaced by the equivalent text prior to being displayed. Example: `"hoverText:$label$newline$amount"`
hoverTextPath:	XPath value	Defines the XPath to the element that provides the hover text for the chart element. The hover text in the XML can utilize all the substitution functionality described above in the **hoverText:** description. Optional.
type:	**bar** **line** **area**	This attribute is used for the **combo** chart type only. It defines how each series on the **combo** chart should be presented.

Parameter	Values	Description
		The following example defines a **combo** chart where one series is rendered as bars and another one as area. `oraChart="type:combo;" oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount;type:bar" oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance;type:area"`
navOpt:	A valid Navigation Option code.	Defines a navigation option to be activated when the chart element is clicked. Optional.
navOptPath:	XPath value	Defines the XPath to a navigation option to be activated when the chart element is clicked. Optional 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.
key:	XPath value	Defines the XPath to an XML element in the series list that contains the key field data to be used in a navigation option. This is required if either **navOpt:** or **navOptPath:** is defined. **NOTE:** Only one key field can be configured for a navigation option.
script:	A BPA script name	Defines a BPA script to be activated when the chart element is clicked. Optional When a script is executed, all the elements from that list item will be made available to the script as temporary variables.
scriptPath:	XPath value	Defines the XPath to a BPA script to be activated when the chart element is clicked. Optional 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.
color:	HTML Color code / RGB value	Optional (default colors applied)

Parameter	Values	Description
		Defines the color for the series. The format is valid HTML color code, e.g. **green** or **blue**. All valid color names are defined in this link: http://www.w3schools.com/htmL/html_colornames.asp. Alternatively an RGB format may be used. (**00FF00** is green and **0000FF** is blue) **NOTE:** Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.
colorPath:	XPath value	Defines the XPath to a color for the series. The valid format as described in the **color:** setting apply. Optional (default colors applied)

pieColors:	HTML color code / RGB values	Defines the colors for the pie series. Any number of HTML color codes or RGB color values can be specified, separated by spaces. Examples:
		`pieColors: green blue`
		`pieColors: 00FF00 0000FF`
		Optional (default colors applied if the series exceeds the values specified)

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.

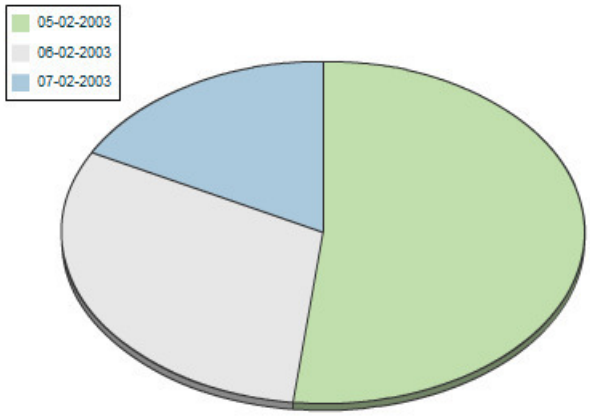
Graph Examples

- 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:

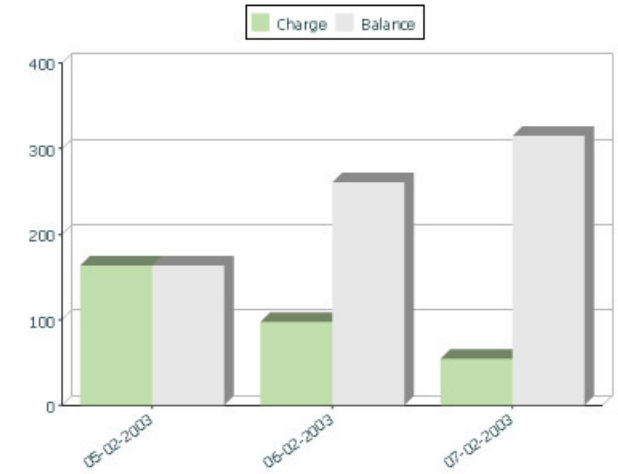


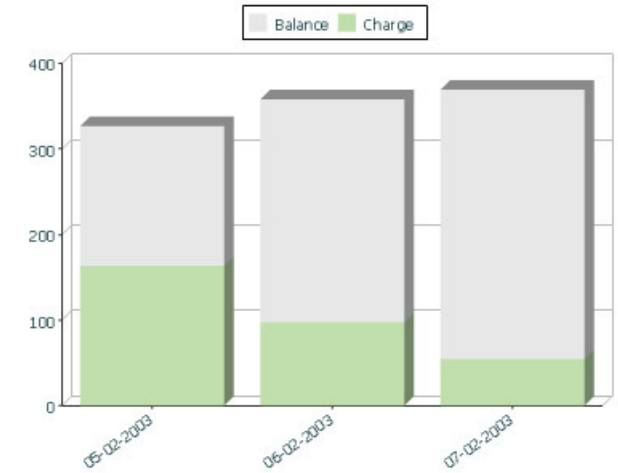
- Sample of a line, cluster, or stacked graph configuration - each with two series:

```
<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.





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. Note that this technique is only recommended for a small to medium volume of data, for example no more than several hundred rows. For higher volumes, it is recommended to use batch upload / download functionality instead.

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.

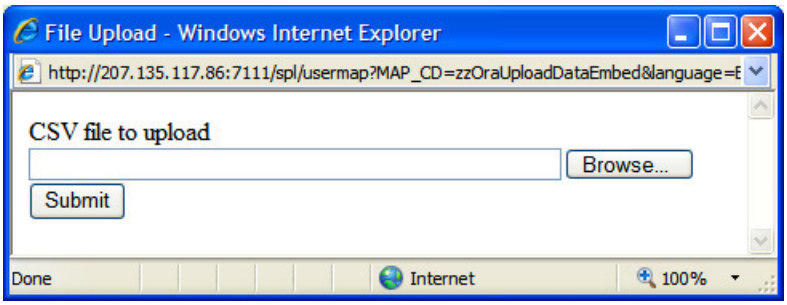
Specifying the optional **showCount:true** value will display the number of records uploaded.

> **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.



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.

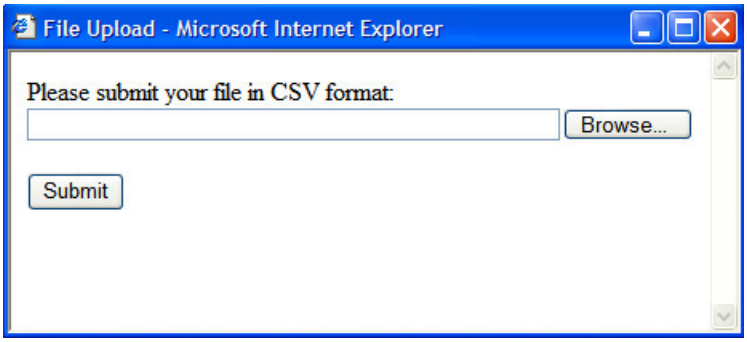
> **CAUTION:** If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to BO Replace Action for more information.

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:

Pressing the "Get Data" button will launch a standard file upload dialogue (provided by Framework) as shown below.



oraDownloadData="list xpath"

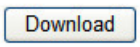
Download configuration requires you to name an XML list to be downloaded. By convention, each list instance will represent a separate row in the created file. By default every element of the list will be comma separated in the file.

> **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.

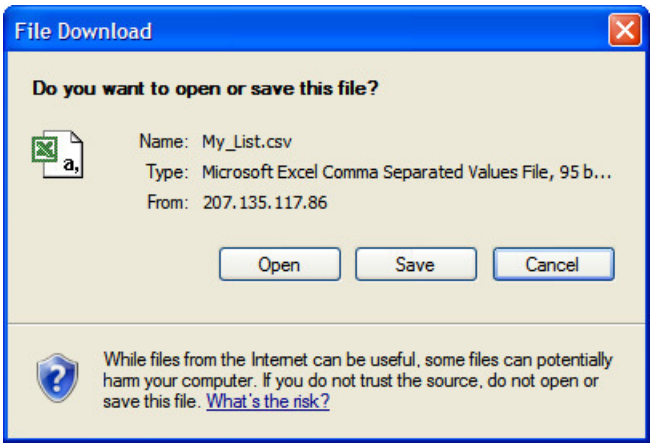
Sample of oraDownloadData.

```
<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>
```

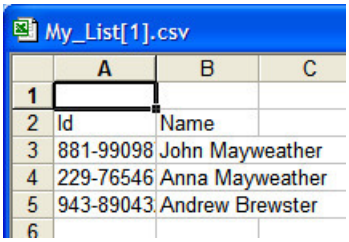
HTML Rendered:



Pressing the "Download" button will launch a standard file download dialogue (provided by Framework) as shown below.



A successful download will result in a CSV file:



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:

Variable	Replacement Logic
[$ZONEDESCRIPTION]	Zone's description text.
[$SEQUENCEID]	Zone's sequence ID.
[$ZONENAME]	Zone's name.
[$HELPTEXT]	Zone's help text.
[$ZONEPARAMNAME]	Zone parameter's value (or blank if it has not been specified).

WARNING:

- Refer to one of the following maps 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)**. Uses the zone's code to retrieve its sequence number.
- **oraIsZoneCollapsed(sequenceId)**. Uses the 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 Load Page Event 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:

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 BO Function which allows for a business object to be invoked within the UI map's HTML. Refer to that section for a description of the first three parameters.

Syntax	Parameters	Comments
oraInvokeBO( )	'BO Name'	
	'XPath' or **null**	
	'action'	
	null	This must be specified as the fourth argument.

Syntax	Parameters	Comments
	[$SEQUENCEID]	This must be specified as the fifth argument.
	true | false	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 BS Function which allows for a business service to be invoked within the UI map's HTML. Refer to that section for a description of the first two parameters.

Syntax	Parameters	Comments
oraInvokeBS( )	'BO Name'	
	'XPath' or **null**	
	null	This must be specified as the fourth argument.
	[$SEQUENCEID]	This must be specified as the fifth argument.
	true | false	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 SS Function which allows for a service script to be invoked within the UI map's HTML. Refer to that section for a description of the first two parameters.

Syntax	Parameters	Comments
oraInvokeSS( )	'Service Script Name'	
	'XPath' or **null**	
	null	This must be specified as the fourth argument.
	[$SEQUENCEID]	This must be specified as the fifth argument.
	true | false	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)
```

Detecting Unsaved Changes

Use this function to return a Boolean set to true if there are unsaved changes. The system will interrogate the function when the user attempts to navigate and issue a warning accordingly. This function is only needed if a UI map is using custom javascript to manage elements such that the system is not able to detect whether changes have been made. Also note that it's the responsibility of the UI map javascript to manage the values in the Boolean used for this function.

```
function hasUnsavedChanges(){
       return isDirtyFlag;
   }
```

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
Using OJET

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 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" 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>
</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="ERRMSG-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('OK');"/>
            </td>
            <td>
              <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button"
               onClick="oraSubmitMap('CANCEL',false);"/>
            </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 cellspacing="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('CANCEL',false);"/>
        </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.	`...`
		` <input class="oraButton"`
		`  oraMdLabel="CANCEL_LBL"`
		`  type="button"`
		`  onClick="oraSubmitMap('CANCEL',false);"/>`
		`...`

oraDisplayCell	Applied to the <td> tag of an embedded table. It defines how the table cell looks (not the data contained inside the cell).	`...`
		`  <td class="oraDisplayCell">`
		`  <span oraField="type"></span>`
		`  </td>`

Style	Comments	Example
		...
oraEmbeddedTable	Applied to the <td> tag that will contain the embedded table.	... <tr> <td colspan="2" class=" oraEmbeddedTable"> <table cellspacing="2"> ... ... </table> </td> </tr> ...
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.	... <table width="100%" cellpadding="12"> <tr class="oraErrorText"> <td> <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"> </td> </tr> </table> ...
oraGridColumnHeader	This style is applied to the <td> tags that define column headers within embedded table.	... <thead > <tr> <th class="oraGridColumnHeaderButton"></th> <th class="oraGridColumnHeaderButton"></th> <th class="oraGridColumnHeader" nowrap="nowrap"> <span oraLabel="contactInformation/type> </th> <th class="oraGridColumnHeader" nowrap="nowrap"> </th> </tr> </thead > ...
oraGridColumnHeaderButton	This style is applied to the <td> tags that define the column headers for the "+" and "-" buttons used on editable embedded tables.	... <thead > <tr> <th class="oraGridColumnHeaderButton"> </th> <th class="oraGridColumnHeaderButton"> </th> <th class="oraGridColumnHeader" nowrap="nowrap"> <span oraLabel="contactInformation/type> </th> <th class="oraGridColumnHeader" nowrap="nowrap"> </th> </tr> </thead > ...
oraInput	This style is applied to input fields: • <input type="text">	... <input type="text" class="oraInput" oraField="address"/> ...

Style	Comments	Example
	• <input type="checkbox"> • <select> • <textarea> **NOTE:** This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".	
oraInputMoney	This style is applied to input fields: • <input type="text"> • <select> (rare) • <textarea> (not recommended) **NOTE:** This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".	``` ... <input type="text" class="oraInputMoney" oraField="amount"/> ... ```
oraInputNumber	This style is applied to input fields: • <input type="text"> • <select> (rare) • <textarea> (not recommended) **NOTE:** This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".	``` ... <input type="text" class="oraInputNumber" oraField="count"/> ... ```
oraLabel	This style is applied to standard label fields that are right aligned. **NOTE:** This can normally be omitted as it is applied by the <col> tag.	``` ... <td class="oraLabel" oraLabel="address"></td> ... ```
oraLabelAlt	This style is applied to standard label fields only if it is desired to have the label aligned to the left.	``` ... <td class="oraLabelAlt" oraLabel="address"></td> ... ```
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.	``` ... <td class="oraLabelCenter" oraLabel="address"></td> ... ```
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.	``` ... <td class="oraLabel"> Google</td> ... ```
oraNormal	This style is applied to standard data fields (display only) that are left aligned.	``` ... <td class="oraNormal" oraField="address"></td> ... ```

Style	Comments	Example
	NOTE: This can normally be omitted as it is applied by the <col> tag.	
oraNormalAlt	This style is applied to standard data fields (display only) only if it is desired to have the data aligned to the right.	... <td class=" oraNormalAlt" oraField="address"></td> ...
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.	... <td class=" oraNormalCenter" oraField="address"></td> ...
oraPageTitle	This style is applied to the element that contains the page title. **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.	... <span class=" oraPageTitle" oraMdField="PAGE_TITLE_LBL"> ...
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. **NOTE:** The style must be applied to both <td> tags or the label may be misaligned with the data/input.	... <tr> <td class="oraSectionEnd" oraLabel="zip"> </td> <td class="oraSectionEnd" oraField="zip"> </td> </tr> ...
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. **NOTE:** The section header should span both the label column and the data column.	... <tr> <td class="oraSectionHeader" colspan="2" oraMdField="DATA_SECTION_LBL"></td> </tr> ...
oraSectionStart	This style is applied to the <td> tags at the start of a "section" (group of elements). It provides some space to separate the section from the previous information (often a section header). **NOTE:** The style must be applied to both <td> tags or the label may be misaligned with the data/input.	... <tr> <td class="oraSectionStart" oraLabel="zip"></td> <td class="oraSectionStart" oraField="zip"></td> </tr> ...
oraTableData	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.	... <colgroup> <col class="oraLabel oraTableLabel"/> <col class="oraNormal oraTableData"/> </colgroup> ...
oraTableLabel	This style is applied to the <col> tag for the label column of the main table (first column). It is used	... <colgroup> <col class="oraLabel oraTableLabel" />

Style	Comments	Example
	to provide a percentage width for the horizontal space to be used for the labels.	<col class="oraNormal oraTableData"/> </colgroup> ...
oraTinyText	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.	... <tr> <td oraLabel="address"></td> <td> <input type="text" oraField="address"/> <div class="oraTinyText" oraField="addressFormatHint"> </td> </tr> ...
oraZoneMap	This style is used applied when the UI Map is to be displayed as a zone on a portal.	... <body class="oraZoneMap"> ...

Using OJET

There are some UI maps delivered by the product that use UI widgets provided by Oracle JavaScript Extension Toolkit (OJET). Releases for OJET do not always align with releases of the framework. In addition, there are times when OJET adjusts APIs that the product uses. The framework will attempt to ensure that each release of the product has the latest and greatest version of the OJET libraries. Implementations are discouraged from attempting to use features in OJET that are not used by the product because the product is not necessarily testing those features and is not ensuring that upgrades to the APIs for those features are backward compatible.

Note that the product isolates the references to OJET into a UI map fragment that is included in the maps that use OJET widgets. This is so that changes to future versions of OJET are minimized to a single place. The map is called F1–OJETLIBS. If your implementation wants to use OJET, the recommendation is to use this UI map fragment.

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 is 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 this value is set to true it disables the generation of unique IDs, thus
replicating the old behavior. When this property is set to false or this property is missing
it enables the generation of unique IDs, thus enabling the list to be standards-compliant.
```

Process Flows

This section describes concepts and provides guidelines related to the configuration of various type of process flows.

Understanding Process Flows

A process flow is a user interface guiding a user through a series of actions in order to accomplish a specific task. The task can be as simple as the collection of information in order to update business data or involve more complex logic such as submitting and tracking batch processes, exchanging messages with an external system, etc.

This section describes topics related to designing and working with process flows.

A Process Flow Is Made Of Panels

Each process flow consist of a number of sequential "steps" needed to accomplish a certain task. Each step is represented by a stop on a progress bar and an associated panel.

Progress is linear in that each step may only lead to a single next step. A process flow always starts at a single initial step but allows for one or more final steps at which the user may choose to complete the process.

At any step the user may take the following actions:

- Enter data or take action as prompted by the step’s panel.
- Continue to a next step, if any.
- Navigate back to any step they have previously visited.
- Finish the process flow if the current step is a final step.
- Save off their work on the process flow and either continue their work or navigate away. At a later time the user may resume their work on the process flow from where they have left off.
- Cancel the process flow.

A process flow type defines the entire metadata needed to control the behavior of process flows of a given type. This includes the sequence of steps, the panel and rules associated with each step and more.

Panel Presentation

The panel presented on each step of the process flow is rendered based on the panel type selected on the step configuration.

For a **Data Area** type of panel, the panel is rendered using UI Hints built into the schema of a specified data area.

For a **UI Map** type of panel, the panel is rendered using a specified UI map fragment.

For a **Panel Set** type of panel, the panel is rendered as a set of tabs, one for each panel that references the current panel as its parent panel. Clicking on each tab renders the UI based on the selected panel’s type. The behavior of a panel set is

analogous to or can be thought of as a nested process flow within a single panel. A panel set may not include another panel set.

A Single UI Map

While the user enters data one panel at a time, data is collected and captured at the entire process flow level. All the elements edited and displayed on the various panels of the process flow must be included in the overall process schema and referenced by their appropriate XPath location in that schema. Each type of process flow defines its unique schema on a designated UI map.

The UI map's HTML should simply include the **Process Flow Controller (F1-ProcessController)** UI map fragment and nothing else. The latter is designed to render the user interface for a process flow based on the metadata defined on its process flow type.

Panel Scripts

A single **Pre-Processing** service script may be associated with a panel for the purpose of preparing the data before the panel is presented to the user. The system calls the script each time the user navigates to this panel on a non-finalized process flow.

A single **Post-Processing** service script may be associated with a panel for the purpose of validating the data entered by the user on that panel. The system calls the script each time the user exits the panel on a non-finalized process flow.

Process Flow Scripts

A single **Process Start** service script may be associated with a process flow for the purpose of preparing the data before a newly initiated process flow is presented to the user.

A single **Process End** service script is executed when the user clicks on the **Finish** or **Cancel Process** buttons to complete the process flow. The entire set of data collected by the process flow UI map schema is provided to this service script for final processing.

NOTE: The schema of these scripts should be the same as the **Process Flow UI map** schema. This would ensure that business rules have access to the entire data captured by the user.

Summary Panel

By default, when a user finishes a process flow, as well as when they view an already closed process, the process flow portal displays the last panel the user was working on. For more complex process flows, you may provide a panel that summarizes information from the entire process. When specified on the process flow type, the system displays that summary panel instead a the default view of a closed process. The user may toggle between the summary view and the detailed view at any time.

The summary panel may be implemented as a UI map fragment or a data area like other panels but may not be associated with any panel script.

Launching A New Process Flow

Each process flow must be initiated and executed by a designated BPA script that has the following simple steps:

- Set BPA area height to zero.
- Populate the temporary variable **$processCode** with the code of the specific process flow type.
- Transfer control to the **Process Flow Navigation (F1-PROCEXEC)** common BPA script.

Launching a process flow of a specific type is assumed to be made from a designated menu option or within a specific context applicable to that type of work. As such, this item needs to be configured specifically for each type of process flow. The navigation option associated with that launching option should reference the BPA script associated with the corresponding type of process flow.

Saving Off Work

The user may save off their work on a process flow at any time. If enabled on the process flow type, the system would also automatically save the data when the user navigates to another panel or away from the process flow portal.

On either manual or automatic save operations, the **Process Manager** script defined on the process flow type is called to save off the entire process flow data on a designated record in the database.

The base product provides a generic **Process Flow** maintenance object that is designed to support the storage of any type of process flow as well a corresponding process manager script. Should process flows of a specific type be stored elsewhere, then a dedicated process manager script should be implemented and used instead.

Review Process Flow Records

A process flow record is created when a user saved off their work or the process flow type enables automatic saving. When the user finishes a process flow, depending on the process flow type configuration the record may be deleted or retained for audit purposes.

Reviewing process flow records of a specific type is assumed to be made from a designated menu option or within a specific context applicable to that type of work. As such this item needs to be configured specifically for each type of process flow. The navigation option associated with that item should reference a query portal designed to query and manage the specific type of process flow records.

In addition to standard query features, the query portal should allow the user to resume their work on a non-finalized process flow as well as review the data captured on a completed record. When implementing such a query portal, refer to the sample **Process Flow Query (F1PRSTRQ)** portal for key features. Your specific product may already include such query portals for process flow types it supports. Refer to your specific product documentation for additional information.

Designing Process Flows

A wide range of process flows may be designed to implement online tasks business users may need to perform. Refer to Understanding Process Flows for more information.

Below is a high-level summary of the steps required to design and configure various types of process flows.

Analyze The Task

The following provides a the high-level guidance in analyzing the necessary data and rules needed to support an implementation of a new type of process flow:

- Identify the data and business rules that are needed to complete the entire task.
  - When possible, organize data into logical groups of details defined as data areas.
  - Design logic for the final processing script that accepts the entire data entered along with the user's action to either complete or cancel the task.
  - If data needs to be initially prepared for a newly initiated process flow, design logic for a corresponding initial processing script.
  - Determine whether a summary panel may be useful for the user to review the outcome of a completed process.
- Design the sequence of panels the user should follow to complete the task. Identify the initial and final panels.
- Review each panel and determine the following:
  - Is it a single panel or a nested process flow (i.e. a panel set)?
  - What is the title of the panel and its label on the progress bar?
  - How should the layout be rendered? Use a data area that leverages UI Hints to describe the panel layout when possible. For more complex HTML requirements you may use a UI map fragment.

    - Should details on this panel be defaulted with some initial values when the user enters this panel? If so, design logic for a corresponding pre-processing script.
    - If data entered on this panel should be validated before the user exits the panel, design logic for a corresponding post-processing script.
- Design the summary panel if needed.
- Determine whether this type of process flow should be automatically saved by the system or solely rely on the user to control when data is saved.
- Determine whether a record of a completed process flow should be retained for audit purposes or deleted upon completion.
- Determine whether a user may delete in progress records of such process flow.
- Determine whether process flows of this type are saved and stored in a dedicated table in the database or are saved in the generic table provided by the base product. The assumption is that introducing new designated storage maintenance objects is not common. Refer to your edge product documentation for additional maintenance objects that may have been provided to support specific types of process flows shipped with the product.
- Determine the context in which a user can initiate, resume work on, and review process flows of that type.
    - The process flow may be launched from a menu or a favorite script etc.
    - Design the query portal a user would use to find and resume a saved off process flow of this type as well as review completed process flow records. Depending on the context of this task, the query may be associated with the same menu line that initiates the process flow.

Using UI Map Panels

Typically a map fragment is rendered as part of a complete UI map but when used as a process flow panel the system renders the panel assuming the fragment contains a complete map.

Therefore, such fragment should includes the following items:

- The standard error message section used for reporting errors.
- This line <xml style="display:none"></xml> at the end of the HTML content for proper binding of data to their HTML elements.

A UI map based panel allows for more control over some of the buttons available to the user. For example, the panel may prevent the user from proceeding to the next panel unless required information is entered. Refer to the explicit APIs listed in the process controller UI Map (F1-ProcessController) for more information.

Configure The Process Flow Type

The following describes the high-level steps needed to configure a new type of process flow:

- Create the UI map for the process flow. Its schema should contain all the details needed to complete the task. Its HTML should only include the process controller UI map fragment.
- Create a start processing service script if needed.
- Create a final processing service script such that it applies all the rules needed to complete or cancel the task.
- For each panel perform the following:
    - Create a UI map fragment for it unless it corresponds to a data area already included in the process flow schema. Make sure the XPath of each element references in the data area or UI map is the same XPath of that element in the process flow’s UI map schema.
    - Create a pre-processing service script if needed. Not applicable to the summary panel.
    - Create a post-processing service script if needed. Not applicable to the summary panel.

    - Set up a label and title MD fields if existing ones cannot be used.
- Create a BPA script for the process flow.
- Configure the process flow type record.
- Set up a navigation option and reference the BPA script on it. Associate the navigation option with a menu or provide other means for the user to initiate the process flow.
- Create a query portal for the specific type of process flow records. Add the new query portal to a menu or provide other means for the user to resume work on a process flow they have saved off or review completed records.

Defining Process Flow Types

This portal is used to maintain process flow types.

Refer to Understanding Process Flows for more information.

You can access the portal from the**Admin Menu** > **System** > **Process Flow Type.**

The following zones may appear as part of the portal's **Main** tab page

- **Process Flow Type List.** This zone lists all process flow types. Broadcast a record to display the details of the selected process flow type.
- **Process Flow Type.** This zone provides information about the selected process flow type.

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 an algorithm. Algorithms are called at strategic points (often referred to as a "plug-in spot") within system processing to perform specific code. The plug-in spots are preconfigured with the product and vary from object to object. Algorithms are used by the base product to provide out of the box functionality. The algorithm framework provides a means for implementations to insert their own custom code, which can be instead of the base delivered code or in addition to it.

NOTE: Algorithm = Plug-in. We use the terms plug-in and algorithm interchangeably throughout this documentation.

Algorithm Type Versus Algorithm

Often the system will just talk about algorithms but in actuality there are two important objects that provide this plug-in functionality.

- An **Algorithm Type** defines the plug-in spot. In addition, it defines program to execute. The program may be defined using Java or using a plug-in script. The algorithm type also defines any parameters that must be supplied to algorithms of this type.
- An **Algorithm** is an instance of an **Algorithm Type**. If the algorithm type does not define any parameters, then there will be only one algorithm for the algorithm type. If the algorithm defines parameters, then there could be several algorithms for a given algorithm type, one for each unique set of parameter values for a given use case. The algorithm is the object that gets referenced on a given control table when configuring your system.

Algorithm Plug-in Spot

The plug-in spot, also called Algorithm Entity and System Event, defines several aspects of the functionality for algorithms that are executed.

- When in system processing are algorithms of this plug-in spot executed? For example, algorithms for the Installation Option - To Do Pre-creation are called when a To Do is getting added, before any insert to the database is performed.
- The API for the plug-in spot. This is often referred to as the "hard parameters" for the plug-in spot. What information is every algorithm provided when called? Additionally, if the algorithm is expected to return information to the caller, this is also part of the hard parameters. For example, the Installation Option - To Do Pre-creation plug-in spot receives all the information about the To Do that is about to be added. It may add or update values. It may also return an indication to not create the To Do, if desired.

  NOTE: The hard parameters are different from the parameters defined on the algorithm type. Those are referred to as the "soft parameters". Those parameters are used to configure settings that the algorithm type code uses per its specific functionality.

- What is the responsibility of algorithms for the plug-in spot? When defining a plug-in spot, the product defines the responsibility of the code for the algorithm types created for that spot. For example, is the algorithm determining information and returning it? Or is it supposed to perform an update? Algorithms for the Installation Option - To Do Pre-creation plug-in spot should not do any updates. Rather they should adjust the To Do information (if desired) in the hard parameters and return them. The calling logic will use that adjusted information when inserting the To Do record.
- Where do you configure the algorithm? We have been using the Installation Option - To Do Pre-creation plug-in spot as an example. As the description suggests, these algorithms are configured on the Installation Option - Algorithm collection. Each plug-in spot defines where (on which control table) the algorithms are plugged in. The system standard for the description of the Algorithm Entity flag value that defines the available plug-in spots is "Control Table Name - Event". For example the To Do Type - Calculate Priority algorithm plug-in spot means that algorithms are configured on the To Do Type. This information also indicates the granularity of the algorithm. Installation Option algorithms are system wide. But an algorithm plugged in on To Do Type, for example, indicates that only To Do Entries for that To Do Type will have that algorithm executed.
- Are multiple algorithms allowed? There are some plug-in spots where it doesn't make sense to support multiple algorithms. An example of this is the Batch Control - Select Records plug-in. This plug-in is used by the plug-in driven batch process and is responsible for selecting the records to process. Only one algorithm of this type is allowed on a given Batch Control. An example of a plug-in spot that would support multiple algorithms is a validation plug-in spot. Note that when a plug-in spot supports multiple algorithms, there can be yet additional aspects of the plug-in spot design.
  - In some cases, this type of plug-in spot simply executes each algorithm. A validation plug-in spot is an example of this.
  - In some cases, the algorithm could return an indication to the plug-in spot 'driver' to not execute any subsequent algorithms. Business Object Status - Enter is an example of a plug-in spot that supports this capability.
  - In some cases, the plug-in spot may support sharing information from one algorithm to the next. Migration Plan - Pre-Compare is an example of a plug-in spot that supports this capability.

Each plug-in spot is defined in the Algorithm Entity lookup (**ALG_ENTITY_FLG**). When defining an algorithm type or defining a plug-in script, once you select the algorithm entity value, a help icon should appear. Clicking this icon will show you the description of the plug-in spot. It may be that online help provides more information about a given plug-in spot as

well. Note that to see the hard parameters defined for a given plug-in spot, go to the Script page in add mode and select the script type of plug-in and the desired algorithm entity lookup value. Then navigate to the Data Area tab. The **View Plug-In Script Data Area** link will display the hard parameters.

Base Product Algorithms

The base product provides many, many algorithm types and algorithms out of the box. Sometimes algorithms are delivered to provide support for common use cases that we know our clients will need. Many algorithms, however, are provided because we are implementing system functionality using the same infrastructure and capabilities that our implementation use for their own custom code.

When an algorithm type is delivered, the decision as to whether an algorithm is also provided depends on some factors related to whether soft parameters are defined for the algorithm type.

- If no parameters are defined for the algorithm type, a corresponding algorithm is typically also provided.
- If parameters are defined and the values are on configuration data that is normally owned by the implementation, a base algorithm will probably not be defined.
- If parameters are defined and the values are not dependent on configuration data, it's possible that one or more algorithms are provided.

To review the algorithm types that are delivered with the product, go to the Algorithm Type query. The query provides several search options, including searching by the algorithm entity lookup value.

How To Add A New Algorithm

Before you can add a new algorithm, you must determine if you can use one of the algorithm types supplied with the system by using the algorithm type query.

If you can use one of the algorithm types, simply add the algorithm and then reference it on the respective control table. Refer to Defining 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 embedded 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.

Defining 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**.

Use the **Algorithm Type Query** portal to search for an existing algorithm type. Once an algorithm type is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Defining Algorithms

If you need to introduce a new algorithm, open **Admin** > **System** > **Algorithm**.

Use the **Algorithm Query** portal to search for an existing algorithm. Once an algorithm is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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.

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.

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 Edit Data Syntax for more information on edit data commands and examples of the use of dynamic data areas.

NOTE: Some server based scripts may not use data areas as means of defining or exchanging data, depending on script type and the chosen scripting technique. Refer to The Big Picture Of Server Based Scripts for an overview of server scripts and their applicable scripting options.

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 for user-invocable BPA scripts. 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. For service scripts, the application service is required.

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

We recommend considering the approaches outlined below when you construct scripts.

While designing your scripts, determine the most maintainable way to set them up. Rather than creating complex, monolithic scripts, we recommend dividing scripts into smaller sections. For example

- 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 using 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 sub script via a **Transfer control** step.

For debugging purposes, 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. There are two techniques you can use to assist in debugging these step types.

- You can instruct the system to display text in the script area.
- 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. This may be used for integrations from an external system or for use cases within the application where a link has been configured to launch the application in a new browser tab, allowing the original page to remain open. Refer to Creating Application Links in External Applications for details of what is supported.

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.

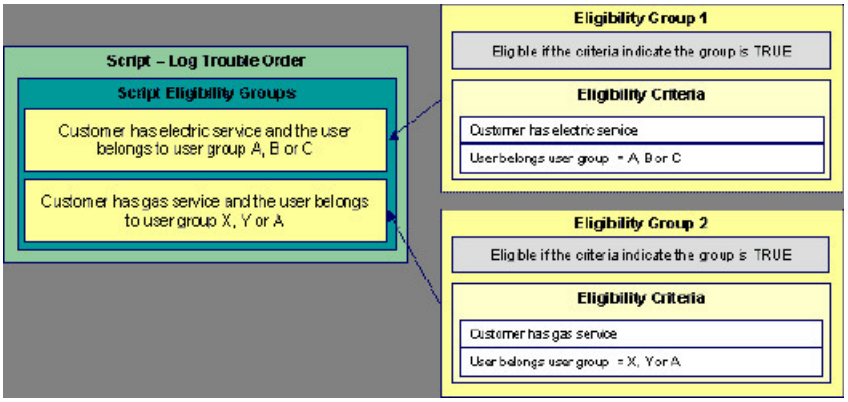
Criteria Groups versus Eligibility Criteria

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**

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).

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). Note that this time span algorithm can be used to compare days, weeks and months as well.

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).

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 looks 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 algorithm type **SECF-USRNGRP** for an example of this type of logic.
- 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 three types of server-based scripts, **Plug-In** scripts, **Service** scripts and **Groovy Library** scripts.

- Plug-in scripts allow an implementation to develop routines that are executed from the system's various plug-in spots. 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. Service scripts are typically written using xpath scripting.
- Groovy Library scripts allow an implementation to develop groups of common routines written in the Groovy language which may be invoked from **Groovy Member** step types.

The topics in this section describe background topics relevant to server-based scripts.

Additional Coding Options For Server Scripts

Server based scripts often perform complex functions best supported by coding in languages with more comprehensive command sets than the base script steps. The system supports two common third-party languages for this purpose.

XML Path Language (XPath) is a language for querying and evaluating elements or nodes in an XML document. XPath commands and expressions can be used directly within Edit Data step types. The script engine version is used to define the applicable XPath version.

Groovy is an object-oriented, dynamic language for the Java platform. The framework supports the use of Groovy within server-based scripts to provide restricted and controlled access to Java-like code, particularly for cloud based implementations. The following topic provides more information on how to incorporate Groovy code into scripts.

Using Groovy Within Scripts

Groovy code can be incorporated in scripts using the step type **Groovy Members**. For each script with Groovy code, there will be a single Groovy class created by concatenating all **Groovy Members** steps.

For security, the product and third party Java classes available for scripting in Groovy will be restricted. The allowable base classes may be viewed via the Groovy JavaDocs feature in the Application Viewer and also via the 'View Groovy Javadocs' link in the context sensitive Script Tips zone. The list of allowable third party classes can be viewed via the 'View third party Groovy whitelist' link in the Script Tips zone.

> **NOTE:** This system supports the use of Groovy for back end processing purposes. It is not intended for user interfaces. Groovy is also not applicable to BPA scripts.

The following describes the two methods for using Groovy.

Using the Scripting Engine

If the script is configured to use a scripting engine version, it can include a mixture of regular and **Groovy Members** step types. The script step types will define the process to be executed. The **Groovy Members** steps will contain code that can be called from **Edit Data** step types within the script using the **invokeGroovy** command. Only Groovy methods that receive no arguments and return void are supported using this command. Refer to the section on edit data steps for more details.

For scripts using this option, the framework provides a superclass containing methods that support common scripting actions such as move commands, string evaluation, and methods to invoke business objects, business services and service scripts. Refer to the Groovy specific JavaDocs for details of the supported methods

Using the Groovy Engine

The system uses an engine version of **Groovy** to indicate that a script is written entirely in Groovy code and can be processed in a similar way to code written in Java. This avoids the need to convert the script to and from XML format and allows the use of code that acts directly on the system objects with consequent performance benefits.

The following script types support the Groovy engine version:

Plug In Scripts

Plug-in Scripts can be configured to use the Groovy engine if they contain only **Groovy Members** step types. The system provides an automatically generated superclass that defines the plug-in spot API. Internally, the Groovy code must conform to the system conventions for Java based algorithm types, including the inclusion of an 'invoke' method that is the plug-in entry point, and the definition of 'soft' parameters using annotations.

Groovy Library Scripts

Groovy Library Scripts provide the ability to create groups of common routines written in Groovy that can be called from within other scripts. Scripts of this type must include a single step type of **Groovy Library Interface** in which the publicly

available methods in the library are listed. The supporting code for those methods is defined in one or more **Groovy Members** step types within the library script. The methods defined in the library can accept arguments and return values of any type. Scripts of this type use the Groovy engine by default and cannot include scripting step types.

Scripts that need to invoke methods from a Groovy library can use the createLibraryScript method provided by the system to instantiate the library interface.

Importing Groovy Classes

The system automatically imports all callable classes available to Groovy at run-time. This includes all allowable domain and white-listed classes. Because it's possible for more than one class with the same name in two different packages, to ensure Groovy uses the class that you intend for it to use in your code, you need to use one of the following techniques:

- Explicitly list the full package name in your code when referencing the class. Example:

  ```
  com.splwg.d1.domain.measurement.measurementServices.common.ConsumptionAmountAndConditionData myAmountAndDat
  ```

- Use the **Groovy Imports** step type to import the class and give the class an alias to then use throughout the code. Example:

  ```
  import com.splwg.d1.domain.measurement.measurementServices.common.ConsumptionAmountAndConditionData as D1Co
  ```

  Any subsequent references to the class can use the alias.

  ```
  D1ConsumptionAmountData myData=...;
  ```

> **NOTE:** Even if you know there are no other classes in your stack with the same name at the time of coding your script, it's possible that in the future a new class is added or class gets added to the whitelist with the same class name.

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.

As an alternative to writing a java program for a plug-in spot, the framework supports creating plug-ins using the defined script steps, Xpath commands, Groovy code or a combination of these three options.

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"

For plug-ins configured to use a script engine version, the best way to understand the 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>
```

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.
- 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.

Plug-ins configured to use the **Groovy** engine version do not use an XML interface for the API and instead are processed in the same way as Java algorithms. The framework supplies a dynamically generated superclass that implements the plug-in spot for Groovy objects. Use the **View Script Superclass** hyperlink to view this superclass and the methods to set and get the hard parameters.

NOTE: For plug-in scripts using the **Groovy** engine version, soft parameters do not appear in the plug-in spot API as defined by the superclass. Plug-ins using only Groovy code define their soft parameters using annotations, in a similar way to Java algorithms, and fetch those values using methods defined in the algorithm code.

Setting Up Plug-In Scripts

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.

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 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 configured to use a scripting engine version 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.

Groovy Library Scripts

Just as service scripts can define common routines written in scripting language, Groovy library scripts are used to define groups of common components or methods written in Groovy. Groovy library code runs on the application server and performs server-based processing for scripts that utilize Groovy code.

The following topics describe basic concepts related to Groovy library scripts.

A Groovy Library Script's API

A Groovy library script's API is composed of one or more public methods whose code is defined in the script's steps. Those methods are defined in a step type of **Groovy Library Interface**. A Groovy library script must have one (and only one) step of this type.

Invoking Groovy Library Methods

Any type of script that supports the **Groovy Members** step type may invoke common methods defined in Groovy library scripts.

Code within a **Groovy Members** step must create an instance of the Groovy library interface definition to enable the interface methods to be invoked. Refer to the topic Using Groovy Within Scripts for more information.

Debugging Server-Based Scripts

The server can create log entries to help you debug your server scripts.

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.

Refer to the Debug Mode topic in the Configuration Tools chapter for details of how to execute the application in debug mode.

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**.

Use the **Script Query** portal to search for an existing script. Once a script is selected, you are brought to the maintenance portal to view and maintain the selected record.

> **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. In addition, the zone provides links to view the Groovy JavaDocs Viewer and the whitelist of third party Groovy classes so that users can verify the restricted list of classes available for Groovy coding in the script.

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**, **Groovy Library 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 a 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. This field does not appear if the script type is **Groovy Library Script**.

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 key information affecting the context and execution of the script.

- Script engine version values of 1, 2 and 3 define the version of the XML Path Language (XPath) to be used for the script. 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.

The default script engine version is 3.0 for plug-in and service scripts. The default version for BPA scripts is 1.0 as higher level versions are not applicable.

There are some additional details to note about script engine version 1.0:

- The XPath library used is Jaxen
- For BPA scripts, it uses the browser's xpath and XML support except for Internet Explorer where the XSXML parser is used.
- 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

- A Script Engine Version value of **Groovy** indicates that only **Groovy Members** step types are used in the script and signals to the system that there is no need to convert the data to and from an XML interface. This allows for greater efficient in script execution. Only plug-in scripts can select the **Groovy** engine version.
- The value **Framework Version 2.1 Compatibility Mode** remains for upgrade purposes. This value should only be applicable to early versions of BPA scripts using syntax that is incompatible with xpath syntax.

NOTE: The **Script Engine Version** field does not appear for **Groovy Library** scripts. The script engine version for these scripts is set to **Groovy** by default and cannot be changed.

Click the **View Script Schema** to view the script's data areas on the schema viewer window. This link does not appear if the script engine version is **Groovy**.

Click the **View XSD** hyperlink to view a script's schema definition in XSD format. This link only appears if the script type is **BPA Script** or **Service Script**.

The **View Script Superclass** hyperlink appears only for plug-in scripts using an engine version of **Groovy**. Click this link to view the code of the runtime generated superclass for the related plug-in spot's implementation.

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 tree summarizes the script's steps. You can use the hyperlink to transfer 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 using a scripting language engine version.

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. Only the field types **Temporary Storage** and **Predefined Value** are applicable when the Condition is **In** or **Between**.

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
Invoking Groovy Code
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" %2='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

Keyword
in
invokeBO
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";** **NOTE:** An XPath expression is surrounded by double quotes.	Move statement with simple XPath reference.	`move "acct/totalBalance" to "parm/formattedValue";`
	Move statement with XPath concatenate function.	`move "concat(person/firstName, ',', person/lastName)" to "parm/fullName";`
	Move statement with XPath substring-before function.	`move "substring-before(parm/fullName,',')" to "person/firstName";`
	Move statement with XPath substring-after function.	`move "substring-after(parm/fullName,',')" to "person/lastName";`
	Move statement with XPath substring function.	`move "substring(parm/date,1,4)" to "parm/year";`
Move to Element **move 'literal' to "xpath";** **NOTE:** A literal value is surrounded by single quotes.	Move statement using a literal string.	`move 'okay for mailing' to "account/preferences[type="mail"]/text";`

Move to Element
move 'Boolean' to "xpath";
	Move statement with Boolean as literal string.	`if ("account/balance > 0")`
		`  move 'true'`
		`  to "account/hasDebitBalance";`
		`end-if;`

	Moving an expression, which results in a Boolean. Note that the filter in the example below is located on a group node.	`<schema>`
		`  <account>`
		`  <hasDebitBalance type="boolean"/>`
		`  <balance/>`
		`  </account>`
		`</schema>`
		
		`move "boolean(account[balance>0])" to "account/hasDebitBalance";`

Move to Group
move "xpath" to "xpath";
	Move a set of elements from one group with another.
	The system matches the initial level of element names and the group / list names from the source schema to the target schema. For lists and groups, the default behavior is to move all elements within the source group / list to the target group / list (even if they are not defined in the target group / list).
	The business service F1-MoveByName may be used for more granular control of the move
		`move "account/custInfo" to "person";`
		
		This statement is equivalent to the following statement:
		
		`move "account/custInfo/*" to "person/*";`

Statement	Example Description	Example Syntax
	(without having to define each individual move statement).	

Move using a Temporary Variable	When moving to a temporary local variable, it is not surrounded by double quotes.
	move "xpath" to $variable;
		`move "count(Person/names/personName)`
		`+ count(Person/ids/personId)"`
		`to $PersonChildCount;`

	When moving from a temporary variable, the variable is surrounded by double quotes. **move "$variable" to “xpath”;**	`move "$AccountBalance"` `to "parm/formattedValue";`

Move using a Context Variable
move "xpath" to $$variable;
move $$variable to “xpath”;
	Context variables, source or target, are referenced without any double quotes.	`move 'context value'`
		`to $$contextVariable;`
		`//`
		`// here, we move from a context variable.`
		`move $$contextVariable`
		`to "MarketMessage/sender";`

Move using a Dynamic Location
move "xpath" | 'literal' to evaluate("xpath" | $variable);
move evaluate("xpath" | $variable) to "xpath" | $variable;
	The evaluate statement allows your move source or target location to be dynamically derived from a variable or schema element location.	`move 'literal'`
		`to evaluate("schemaLocation");`
		`//`
		`move "schemaLocation"`
		`to evaluate($Variable);`
		`move evaluate("schemaLocation")`
		`to $Variable;`
		`//`
		`move evaluate($Variable)`
		`to "schemaLocation";`

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.
		`move escape("schemaLocation")`
		`to $Variable;`
		`//`
		`move escape($Variable)`
		`to "schemaLocation";`

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:
		`if ("boolean(customer/securityEnabled)")`
		`  goto updateInfo;`
		`else`
		`  move null to "customer";`
		`end-if;`

Statement	Example Description	Example Syntax

		Remove all child nodes of a group node with the suffix '/*'.
		`if ("boolean(customer/securityEnabled)")`
		`  move null to "customer/*";`
		`end-if;`

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. This option is better than add for performance purposes if there is no reason to re-read the record.
- **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. This option is better than update for performance purposes if there is no reason to re-read the record.
- **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.

NOTE: The replace action is currently not supported for any maintenance object that is maintained in a 'fixed' page that uses a list metaphor to show all the records in the page at once. Currency is an example of this type of page.

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";
```

BO Warnings. Note that for BPA scripting, the **invoke** statements may also indicate how to handle warnings.

Syntax	Description	Examples
with warn asError	Indicates that a warning should be treated as an error displayed in the UI map. The text **asError** is optional.	`invokeBO 'BusinessObject' using "dataArea" for add` `invokeSS 'ServiceScript' using "dataArea" with war`
with warn popup	Indicates that a warning should be presented in the standard framework popup. In this scenario, the user is presented with standard OK and Cancel buttons. If the user clicks OK, it means that the process should continue. If the user clicks Cancel, the processing should discontinue. This is the recommended setting.	`invokeBS "daName/ bsElement" using "dataArea" with warn popup;`
with warn suppress	Indicates that a warning should be suppressed. This is the default if no warning syntax is added to the **invoke** statement.	`invokeBS "daName/ bsElement" using "dataArea" with warn suppress;` `invokeSS 'ServiceScript' using "dataArea";`

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**.

For BPA scripts, there should also be logic following the invocation to handle errors and warnings (if **with warn as popup** is used). The system variable **$WARNING** is set to **true** if the user has clicked the Cancel button in the warning popup. If a map was shown, the logic should redisplay the map (where the warning message will be redisplayed.) This allows the user to potentially make changes and save again. If no map was shown prior to the warning, then the logic should terminate.

The system variable **$ERROR** indicates that an error was received. If a map was shown prior to the error, then the logic should redisplay the map where the error will be shown. If no map was shown, then the product provides a BPA Script **F1–HandleErr** that should be used to display the error. The following is an example of typical error handling logic.

```
invokeBO "F1-DetermineBo/output/bo" using "boSchema" for update with warn popup;
```

```
if ("$WARNING")
   if ("map_schema/action = 'DEL'")
     terminate;
   else
     goto maintMap;
   end-if;
end-if;
if ("$ERROR")
  if ("map_schema/action != 'DEL'")
    goto maintMap;
  else
    transferControl 'F1-HandleErr';
  end-if;
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.	`if ("string(parm/spouse/name) != $BLANK")`
		`  goto addSpouse;`
		`end-if;`

$CURRENT-DATE	Represents the current date. For BPA scripts, this is the browser date. For server scripts this is the server date (and is affected by the system date override logic).	`move "$CURRENT-DATE" to $tempDate;`
$CURRENT-STD-DTTM	Represents the current date-time expressed in standard time (meaning without any adjustments for summer time / daylight savings time).	`move "$CURRENT-STD-DTTM" to $tempDateTime;`
$DEVICE-OS	Represents the user's device operating system.	`move "$DEVICE-OS" to $tempDeviceOs;`
$DEVICE-BROWSER	Represents the user's device browser.	`move "$DEVICE-BROWSER" to $tempDeviceBrowser;`
$DEVICE-DISPLAY-TYPE	Represents the user's device screen display type whether it is Desktop size or Medium or Small size. Returned values may be like oraDesktop, oraTablet and oraPhone.	`move "$DEVICE-DISPLAY-TYPE" to $tempDeviceDisplayType;`
$DEVICE-INFO	Provides the combination of all three device properties (DEVICE-OS, DEVICE-BROWSER and DEVICE-DISPLAY-TYPE) and each property value is separated by semi-colon.	`move "$DEVICE-INFO" to $tempDeviceInfo;`

System Variables - BPA Scripts Only

The following system variables are only available / applicable for BPA script types.

Variable	Description	Example
$DOUBLE_QUOTE	Represents a double quote.	`move "$DOUBLE_QUOTE" to $tempField;`
$SINGLE_QUOTE	Represents an apostrophe.	`move "$SINGLE_QUOTE" to $tempField;`
$SPACE	Contains a single space value.	`move "$SPACE" to $tempField;`
$SYSTEM-DATE	Represents the server date. Note that this date is affected by the system date override logic)	`move "$SYSTEM-DATE" to $tempDate;`

System Variables - Server Scripts Only

The following system variables are only available / applicable for service script and plug-in script types.

Variable	Description	Example
$ADDITIONAL-IP-INFO	An HTTP request includes an "additional IP address" header field. This may be populated by an implementation if there is some information available on the proxy server or load balancer, such as the originating IP address.	`move "$ADDITIONAL-IP-INFO" to "parm/request/headerIpAddress";`
$CURRENT-DTTM	Represents the current date-time.	`move "$CURRENT-DTTM" to $tempDateTime;`
$F1-INSTALLATION-TIMEZONE	Represents the time zone code defined on the installation options.	`move "$F1-INSTALLATION-TIMEZONE" to $timeZone;`
$LANGUAGE	Represents the language code the script is using. Typically this is the user's default language.	`move "$LANGUAGE" to $tempLanguage;`
$PROCESS-DATE	Represents the process date. The process date differs from the current date because the process date will remain consistent throughout the duration of the process being executed. For example, if a service script stores several business objects – the process date is initialized at the start of the service script execution and each business object will have the same process date defaulted. The current date, especially the current date time, will reflect the actual time of processing.	`move "$PROCESS-DATE" to $tempDate;`
$PROCESS-DTTM	Represents the process date-time. Note that the process date and time is initialized at the start of a particular process and will not reflect the exact date and time of an update.	`move "$PROCESS-DTTM" to $tempDateTime;`
$REQUESTING-IP-ADDRESS	Represents the IP address from the HTTP request. Note that if the request is routed through a proxy server or load balancer, this IP address is be the IP address of the proxy or load balancer, not the IP address of the end user. Refer to the **$ADDITIONAL-IP-INFO** variable for information.	`move "$REQUESTING-IP-ADDRESS" to "parm/request/systemIpAddress";`
$USER	Represents the user ID of the user executing the script.	`move "$USER" to $tempUser;`

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.

Syntax	Valid Values	Comments
performScript	'BPA Script Name'	Script to perform is explicitly provided.
	$Variable	Script to perform is found in a variable.
	"XPath"	Script to perform is found in an element, referenced by its XPath.
transferControl	Analogous to the **performScript** statement	

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.

Syntax	Valid Values	Comments
navigate	'Navigation Code'	Navigation option is explicitly provided.
	$Variable	Navigation option is found in a variable.
	"XPath"	Navigation option is found in an element, referenced by its XPath.

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.

Syntax	Valid Values
navigateAndReloadDashboard	Analogous to the **navigate** statement

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.

Syntax	Valid Values	Comments
declareBOWithBOGroup	'BO Name'	BO is explicitly provided.
	$Variable	BO is found in a variable.
	"XPath"	BO is found in an element, referenced by its XPath.

The following table highlights additional syntax for this statement.

Syntax	Valid Values
as	'Dynamic Schema Name'

Examples:

```
declareBOWithBOGroup 'BusinessObject' as 'newMapSchema';

declareBOWithBOGroup $variableBO as 'newMapSchema';

declareBOWithBOGroup "daName/boElement" as 'newMapSchema';
```

Invoke Map Statement

The edit data step supports functionality analogous to the Invoke map step type. This is applicable only to BPA scripts.

Syntax	Valid Values	Comments
invokeMap	'Map Name'	UI Map is explicitly provided.
	$Variable	UI Map is found in a variable.
	"XPath"	UI Map is found in an element, referenced by its XPath.

The following table highlights additional syntax for this statement.

Syntax	Valid Values	Comments
using	"Data Area group name"	Indicates the data area to be passed to and from the server when rendering the HTML form associated with the map.
target	bpa	
	page	
	popup	

Refer to the Invoke map step type for more information about the **target** values.

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;
```

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.

Syntax
generateBOEditMap
generateBSEditMap
generateSSEditMap
generateDAEditMap

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 following table highlights additional syntax for this statement.

Syntax	Valid Values	Comments
using	"Data Area group name"	Indicates the data area to be passed to and from the server when rendering the HTML form associated with the map.
target	bpa	
	page	
	popup	

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.

Syntax	Attributes	Comments
terminate with error (x, y %n= element=)	'x' represents the message category	Required.
	'y' represents the message number	Required.
	%n="Element XPath" or %n='literal'	Specify the substitution parameters supported by the message using either literal values or XPath references.
	element='Element XPath' or element= $variable	Optionally specify an element name within a UI map to highlight as part of the error. When the element in error is within a list, you must populate a variable with the XPath information, including the list occurrence and use the variable in the element reference.

Example of a simple field:

```
if ("string(customer/lastName) = $BLANK")
  terminate with error (8000, 1001 %1="customer/
lastName" %2='Last name required' element='customer/lastName');
end-if;
```

Example of terminate where the element to mark is in a list:

```
move "0" to $count;
default $elementReference;
for ($list in "parm/hard/newBusinessObject/listName")
  move "1 + xs:integer($count)" to $count;
  if //** check some condition for elementName
     move "concat('list[',$count,']/elementName')" to $elementReference;
      terminate with error (11000, 11000 element=$elementReference);
  end-if;
end-for;
```

FASTPATH: For more information on presenting errors in a UI map, please refer to Display Errors.

Invoking Groovy Code

The edit data step supports the ability to invoke Groovy code using the syntax **invokeGroovy 'method';** where 'method' is the name of a method defined in a **Groovy Members** step within the script. Only methods that do not receive arguments and return void can be invoked in this way. However, the method invoked from the edit data step can be supported by additional Groovy code in other **Groovy Members** step types.

Example Edit Data step:

```
invokeGroovy 'invoke';
```

Example Groovy Members step:

```
void invoke() {
  initParms()
  readBO()
  initConfig()
  retrieve()
  updateBO()
}
```

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 and Above Notes
Scripting using the engine version 2 or above 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 OUAF
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 $latestMsrmtYear;
```

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 $qnty;
move "currency/decimals" to $decimals;
move "fn:round(xs:decimal($qnty) * 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 know 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(D1-UnitOfMeasure/
measuresPeakQuantity) = 'D1MP') then 'D1MX' else 'D1SM' " 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 versions 2 and above support XQuery 1.0 Functions and Operators, and the 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. Refer to BO warnings for more information about handling warnings when invoking a BO.

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**.

> **FASTPATH:** Refer to BO Actions for more information about the various actions.

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.

Step Types Applicable to Server Based Scripts only

The contents of this section describe step types that are only applicable to server based scripts.

How To Set Up Groovy Member Steps

Groovy Member steps provide a free format text area where you can enter Groovy code.

Enter a description of the code block in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter your code in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for editing the code.

NOTE: While it is possible to set up multiple steps of type **Groovy Member** the system treats these steps as a single class for compilation and execution purposes. Refer to the topic Using Groovy Within Scripts for more information.

How To Set Up Groovy Imports Steps

Groovy Imports step types provide a free format text area where you can list the Groovy classes to be imported for use by the code in **Groovy Member** steps within the script.

Enter a description of the imports step in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter the list of classes to import in the **Edit Data Text** field using the syntax **import 'class';** where 'class' is the fully qualified package name of the Groovy class. Click the adjacent icon to open a window providing more space for editing the text.

> **NOTE:** For security, the classes that may be imported are restricted to those allowed by the Framework. Refer to the topic Using Groovy Within Scripts for more information.

How To Set Up Groovy Library Interface Steps

Groovy Library Interface steps are only applicable to **Groovy Library Scripts**. They provide a free format text area where you can list the Groovy methods defined within the script that are available for use by other scripts.

Enter a description of the interface step in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter the list of interface methods in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for editing the list.

> **NOTE:** Every **Groovy Library Script** must have one and only one step of type **Groovy Library Interface**. The supporting code for the available methods is defined using one or more **Groovy Member** steps in the same script. Refer to the topic Using Groovy Within Scripts for more information.

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:

Button Name
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:

Button Name
IM_SAVE
IM_REFRESH
IM_CLEAR
IM_COPY
IM_DELETE
IM_ScrollBack
IM_ScrollForward

The following buttons are also supported:

Button Name	Comments
IM_TO_DO	
IM_PrevTo Do	This brings you to the To Do Summary page.
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. Note that when a script is finished, it will return the dashboard to the state it was when the script was launched.
IM_MAXIMIZE_DASHBOARD	Pressing this will expand the dashboard. Note that when a script is finished, it will return the dashboard to the state it was when the script was launched.

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 "PressContinue after you've selected the customer" (the **** and **** 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.

NOTE: Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.

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:

Variable Name	Comments
%PARM-<name>	This 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.
%PARM-NOT-SET	This is to be used to compare against **%PARM-< >** parameters to check if the parameter has been set or not when the application was launched. A parameter that has not been set would be considered equal to this global variable. It is recommended to compare parameters against this global variable before using them for the first time.
%BLANK	A constant that contains a blank value (no value).
%SPACE	A constant that contains a single space value.
%CURRENT-DATE	The current date as known by the browser, not the server.
%SYSTEM-DATE	The server date. Note that this date is affected by the system date override logic)
%SAVE-REQUIRED	A flag that contains an indication of whether the data on a page has been changed (and thus 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	A constant that contains a new line character (carriage return). Upon substitution, a line break is inserted in the resultant text. **NOTE:** This constant 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:

Variable Name	Comments
%CONTEXT-PERSONID	A constant that contains the ID of the current person.
%CONTEXT-ACCOUNTID	A constant that contains the ID of the current account.
%CONTEXT-PREMISEID	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:

Variable Name	Comments
%ERRMSG-CATEGORY %ERRMSG-NUMBER	The unique identifier of the error message number.
%ERRMSG-TEXT	The brief description of the error.
%ERRMSG-LONG	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_FWDTTM_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 **MSGPARM[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 **MSGPARM[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 notions. Use the **View Script Schema** hyperlink provided on the Script - Step tab page to launch the script's data areas schema.

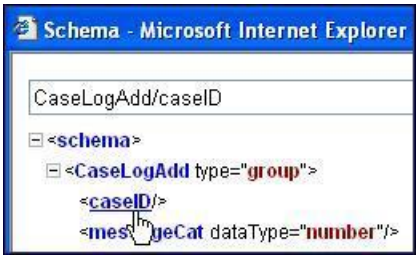


Figure 4: Schema Viewer

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.

> **NOTE: Conditional tab page.** This tab page does not appear for Groovy Library scripts or plug-in scripts using the **Groovy** engine version.

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 only for plug-in scripts using a script engine version. 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.

- 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.

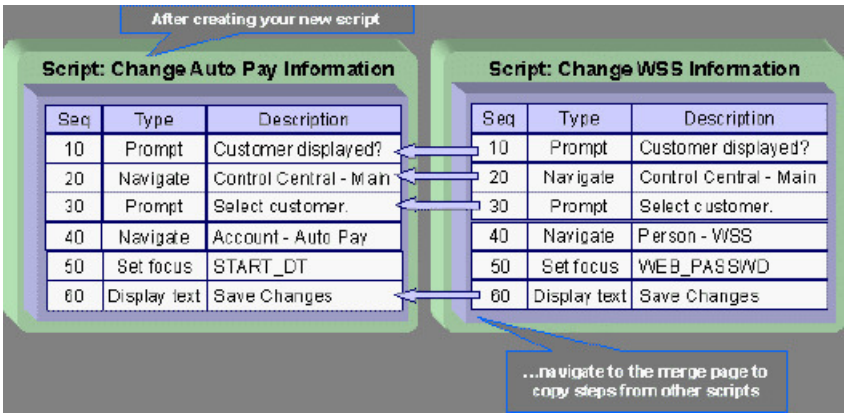
- Use **Criteria Comparison** to define the method of comparison:
  - Choose **Algorithm** if you want an algorithm to perform the comparison and return a value of True, False or Insufficient Data. Push the adjacent search button to select the algorithm that is responsible for performing the comparison.
  - 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.

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:

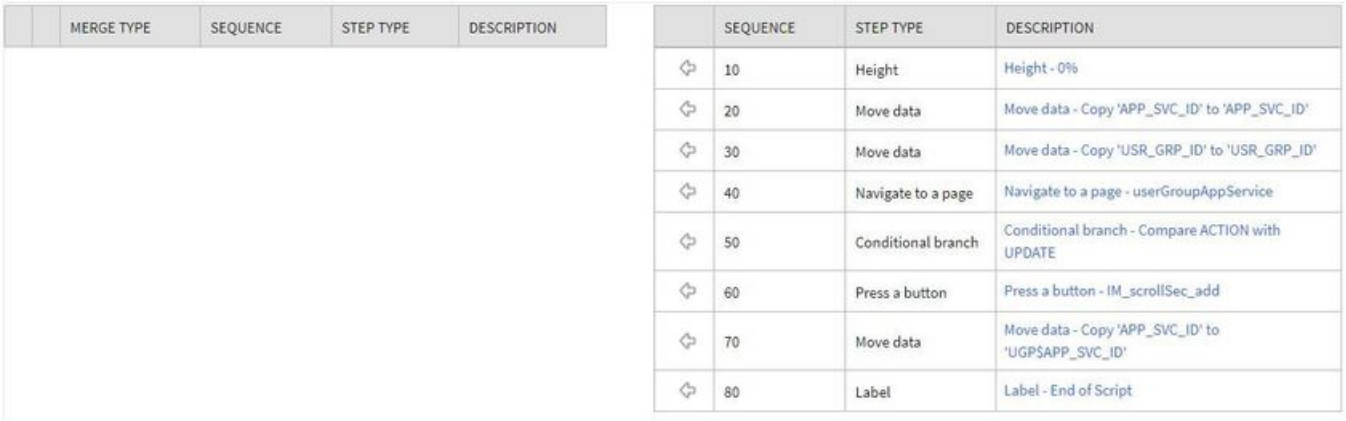
		MERGE TYPE	SEQUENCE	STEP TYPE	DESCRIPTION
		Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
		Original	20	Edit data	Edit data - Read the Bundle
		Original	40	Edit data	Edit data - Create new bundle XML

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.

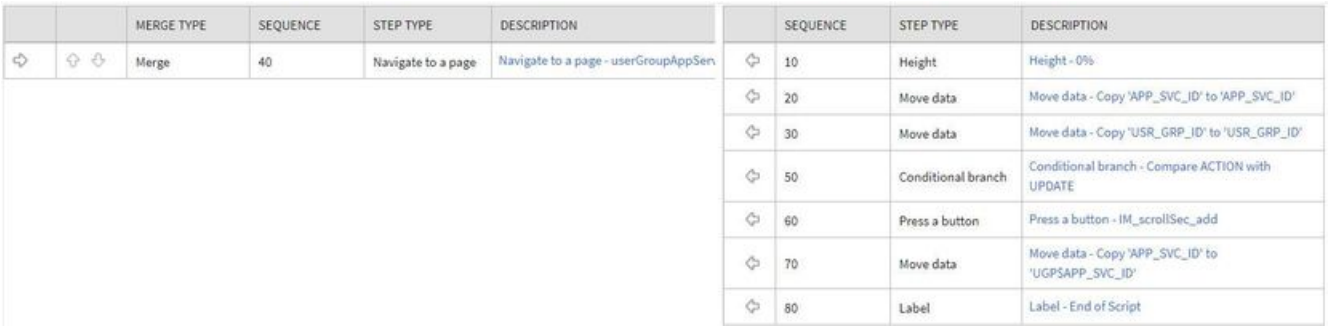
Adding a Step to a Script

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.



The step is moved to the left portion of the page.

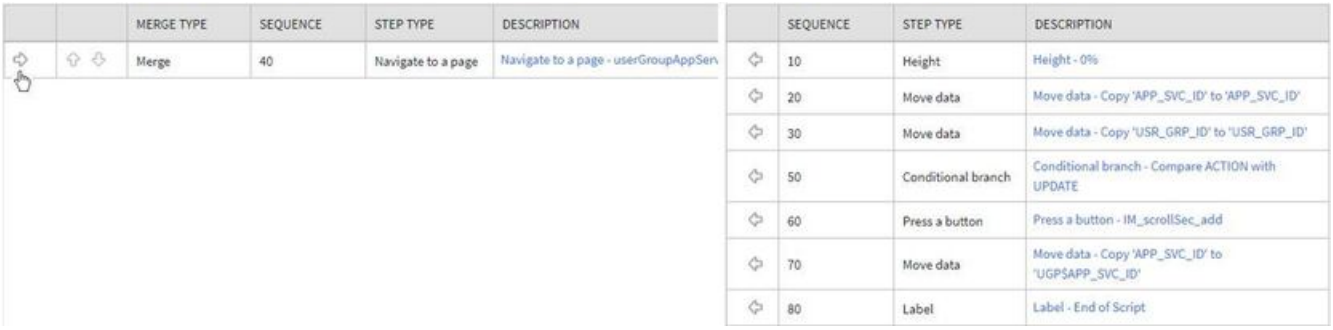


		MERGE TYPE	SEQUENCE	STEP TYPE	DESCRIPTION
		Merge	40	Navigate to a page	Navigate to a page - userGroupAppSer

	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'
	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 a page - userGroupAppSer

	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'
	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: Only page services can be invoked.

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.

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: Fields in multiple lists. 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. 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']**.

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**.

Use the **Managed Content Query** portal to search for an existing managed content entry. Once the managed content is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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	

Mnemonic	Valid Values	Description	Examples
		field element, unlike group or list, will contain information in its nodes - rather than other nodes.	

	"group"	The group element is typically a structural element of the schema only, in which case it has no mapping.
		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.
			Example where a group is used to create a structure
			`<schema>`
			`  <input type="group"`
			`  <userId/>`
			`  </input>`
			`  <output type="group">`
			`  <firstName/>`
			`  <lastName/>`
			`  </output>`
			`</schema>`
			Example where the group includes the mapping:
			`<parameters type="group" mapXML="BO_DATA_AREA" mdField= "F1_TODOSUMEMAIL_PARM_LBL"/ >`
			`  <numberOfDays`
			`  mdField="F1_NBR_DAYS"`
			`  required="true"/>`
			`  <frequency`
			`  mdField="F1_FREQUENCY"/ >`

	"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.	Example of a schema with a list:
			`<schema>`
			`  <statesList type="list">`
			`  <state isPrimeKey="true"/ >`
			`  <description/>`
			`  </statesList>`
			`</schema>`
			Example of a schema with a list:
			`<xml>`
			`  <statesList>`
			`  <state>AK</state>`
			`  <description>Alaska</ description>`
			`  </statesList>`
			`  <statesList>`
			`  <state>AL</state>`
			`  <description>Alabama</ description>`
			`  </statesList>`
			`...`
			`</xml>`

	"raw"	The raw data type is used to capture a chunk of raw text that doesn't have any inherent structure associated with it.	`<sendDetail type="raw" />`
			Example of an XML instance for the above schema:
			`<sendDetail>`
			`  <messageInfo>`
			`  <senderAddress>123 W. Main St,  On`
			`  </senderAddress>`
			`  <corpZone>3A</ corpZone>`
			`  </messageInfo>`
			`</sendDetail>`

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.	`<schema>`
			`  <custName dataType="string"/`
			`>`
			`</schema>`

	"number"	Defines an element that is a number.
		
		NOTE: UI hints include a setting to Suppress Automatic Number Formatting.
		
		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.
			Examples
			`<schema>`
			`  <count dataType="number"/`
			`>`
			`</schema>`
			
			`<schema>`
			`  <taxRate dataType="number"`
			`   currencyRef="currency"/`
			`>`
			`</schema>`

	"money"
	Optional additional attributes
	currencyRef="element name"
		Defines an element that represents a monetary amount.
		
		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.
		
		NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.
			`<schema>`
			`  <currency default="USD"`
			`    suppress="true"/>`
			`  <balance dataType="money"`
			`   currencyRef="currency"/`
			`>`
			`</schema>`

	"lookup"
	Required additional attribute
	lookup="field name"
		Defines an element that has valid values defined using a lookup. The lookup field is required.	`<schema>`
			`  <status dataType="lookup"`
			`   lookup="STATUS_FLG"/>`
			`</schema>`

	"lookupBO"
	Required additional attribute
	lookupBO="bo name"
		Defines an element that has valid values defined using an extendable lookup. The	`<schema>`
			`  <category dataType="lookupBO"`
			`   lookupBO="CM-`
			`BusinessCategory"/>`
			`</schema>`

Mnemonic	Valid Values	Description	Examples
		extendable lookup's business object is required.	
	"boolean"	Defines an element that has values of "Y" and "N".	`<schema> <allowsEdit dataType="boolean"/ > </schema>`
	"date"	Defines an element that represents a date.	`<schema> <startDate dataType="date"/ > </schema>`
	"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.	`<schema> <startDateTime dataType="dateTime"/> </schema>`
	"time"	Defines an element that represents a time.	`<schema> <startTime dataType="time"/ > </schema>`
	"uri"	Defines an element captures a URI. Elements defined with this type enable the support for URI Whitelist and Substitution as described in Referencing URIs .	`<schema> <exportDirectory dataType="uri"/ > </schema>`

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. Refer to Designing Time Zones for more information.

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.	`<schema>`
			`  <factId`
			`  mapField="FACT_ID"/>`
			`</schema>`

mapChild=	"table name"	The **mapChild** attribute is used only for business object mapping. It is used in two ways:
		• 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.
			Example of a list within a child table in a BO:
			`<persons type="list"`
			`  mapChild="CI_ACCT_PER"`
			`  <personId mapField="PER_ID"/`
			`>`
			`</persons>`

mapList=	"list name"	The **mapList** attribute is used only for business service mapping. It is used in two ways:
		• 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 that a flattened field lives in. For more information on flattening, refer to flattening section below.
			Example of a list within a business service:
			`<selectList type="list"`
			`  mapList="DE"`
			`  <value mapField="COL_VALUE">`
			`  <row mapList="DE_VAL">`
			`  <SEQNO is="2"/>`
			`  </row>`
			`  </value>`
			`</selectList>`

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 the structure. It is also possible to map list elements to a large character field associated with the list child.	`<enrollmentRequest type="group"`
			`mapXML="CASE_CLOB">`
			`  <messageID/>`
			`  <sender/>`
			`</enrollmentRequest>`
			`<enrollmentKey`
			`mapXML="CASE_CLOB"/>`
			`<enrollmentInfo type="list"`
			`mapChild="CI_CASE_CHILD">`
			`  <sequence`
			`  mapField="CHILD_SEQ"/>`
			`  <name`
			`  mapXML="CASE_CHILD_CLOB"/`
			`>`
			`  <value`
			`  mapXML="CASE_CHILD_CLOB"/`
			`>`
			`</enrollmentInfo>`

isPrimeKey=	**"true"**	You must specify a primary key for a list defined within a mapped	`<questionnaire`
			`type="list" mapXML="CASE_CLOB">`
			`  <question`
			`  isPrimeKey="true"/>`

Mnemonic	Valid Values	Description	Examples
		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.	<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 (ascending is the default). **NOTE:** This 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.

Mnemonic	Valid Values	Description	Examples
<!-- comment -->		Use this to add a comment to a schema by using special open and close characters: <!-- and -->.	<schema> <!-- This schema is used to capture business information only, please refer to the 'HUMAN' BO for person information --> </schema>
description=	"text"	The description of an element may be used to provide 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	<schema> <active type="boolean" description="active account" label="Active"/>

Mnemonic	Valid Values	Description	Examples
		that would typically precede the element in a user interface.	</schema>
required=	**"true"**	Used to require the existence of an element during object interaction. **NOTE:** For included schemas, the required="true" attribute is not 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=	"field code"	The meta-data field attribute is used to associate an element with a field's metadata. The field defines data type, as well as its label and help text. If you link an element with a meta-data field, you don't need to specify any of these attributes. **NOTE:** For a business object schema, the mapField attribute is used to derive the data type and label. An mdField attribute may be provided to override these attributes, if needed. If the element is mapped to an XML column, the mdField is needed to provide the appropriate data type and label.	<schema> <active mdField="CM_ACTIVE_SW"/> </schema>
fkRef=	"FK Reference Code"	If the element is a foreign key, defining its FK Reference will enable framework validation of the element during schema interaction and automatically provide descriptions and navigation capability.	<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.	<schema> <type mdField="SA_TYPE_CD" default="E1" private="true"/ > </schema>

Mnemonic	Valid Values	Description	Examples
		NOTE: A private element requires a default.	

suppress=	**"true"**	This setting prevents an element from appearing in automatically generated user interfaces.
		NOTE: This attribute can be specified on a group, in which case all elements of the group will be suppressed.
			`<schema>`
			`  <ls mdField="LIFE_SUPPORT_FLG" default="N" suppress="true"/`
			`<`
			`</schema>`

	"blank"	This setting means that automatic UI rendering 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>`

	"input"	This setting means that automatic UI rendering will suppress the element for input, although it may still be displayed if it is 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"**	This setting prevents an element from appearing as a changed element in the business object's audit plug-in spot. If specified on a group or list node it applies to the whole node. You cannot specify it on the schema root node, only on schema elements.
		NOTE: This attribute is only applicable to business object schemas.
			`<schema>`
			`  ...`
			`  <version mapField="VERSION_NBR" noAudit="true"/>`
			`  ...`
			`  <workFields type="group" noAudit="true"`
			`  <lastProcessedId/>`
			`  <lastProcessedTime/>`
			`  </workFields>`
			`</schema>`

storeEmptyNodes=	**"true"**	By default, empty nodes are removed from a business object instance when saved. This setting allows a group or a list element to explicitly keep its empty nodes.	`<schema>`
			`  <message type="group" storeEmptyNodes="`
			`  ...`
			`  </message>`
			`</schema>`

Mnemonic	Valid Values	Description	Examples
		This attribute may be useful in situations where business object data is exchanged with an external system and there is a need to distinguish between an empty value for an element that participates in the synchronization and an element that does not participate and therefore omitted from the message altogether. **NOTE:** This is only available for group and list elements.	

emitEmptyGroups=	**"true"**	By default, empty nodes are not included in the output of a business object or a business service call. This setting allows a group or a list element to explicitly request its empty nodes to be included in the output.
		
		This attribute may be useful in the same situations that may required the use of the **storeEmptyNodes=** attribute.
		
		NOTE: This is only available for group and list elements on business object and business service schemas.
			`<schema>`
			`  <message type="group" emitEmptyGroups="`
			`  ...`
			`  </message>`
			`</schema>`

emitEmptyElements=	**"true"**	This attribute is similar to the **emitEmptyGroups=** attribute but may also be specified at the field level. The attribute is applicable to all schema types.	`<schema>`
			`  <messageNumber type="field" emitEmptyE`
			`  ...`
			`  </message>`
			`</schema>`

adheresToDA=	“comma separated list of data area names.”	This attribute is applicable to the schema, group and list nodes. It claims that this sub-schema node is intended to "look like" the schema of one of the listed data areas, for polymorphism reasons in general.
		
		NOTE: The recommended approach for enforcing the structure of a shared data area is by including it in the schema. This attribute should be limited to use cases where the ability
			`<schema>`
			`  <messageNumber type="group" adheresToDA`
			`  ...`
			`  </message>`
			`</schema>`

Mnemonic	Valid Values	Description	Examples
		to include the data area is not possible.	

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 a field is displayed on the user interface in Add mode, the default value defined in the schema is shown. In addition, the server logic uses the default value if no value is supplied and the element is marked as required or suppressed.

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.	`<schema>`
			`  <perType mapField="PER_OR_BUS_FLG" default="P" required="true"/>`
			`</schema>`
			
			`<schema>`
			`  <frequency dataType="number" default="1" required="true"/>`
			`</schema>`

	"%CurrentDate"	Used to default the element to the current date. This is only applicable to date elements.	`<schema>` `  <logDate mapField="LOG_DT" default="%CurrentDate" required="true"/>`

Mnemonic	Valid Values	Description	Examples
			</schema>
	"%CurrentDateTime"	Used to default the element to the current date / time. This is only applicable to date / time elements.	<schema> <logDateTime mapField="LOG_DTTM" default="%CurrentDateTime" required="true"/> </schema>
	"%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.	<schema> <startDateTime mapField="START_DTTM" default="%StandardDateTime" required="true"/> </schema>
	"%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 to the batch business date that is a standard batch parameter.	<schema> <billDate mapField="BILL_DT" default="%ProcessDate" required="true"/> </schema>
	"%ProcessDateTime"	This is similar to **"%ProcessDate"** but for date / time fields.	<schema> <calcDateTime mapField="CALC_DTTM" default="%ProcessDateTime" required="true"/> </schema>
	"%CurrentUser"	Used to default the element to the current user.	<schema> <logUser mapField="LOG_USER" default="%CurrentUser" required="true"/> </schema>
	"%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.	<schema> <timeZone default="%CurrentUserTimeZone" required="true"/> </schema>
	"%CurrentUserLanguage"	Used to default the element to the current user's language.	<schema> <custLanguage mapField="CUST_LANG" default="%CurrentUserLanguage" required="true"/> </schema>
	"%InstallationCurrency"	Used to default the currency from the installation record.	<schema> <currency mapField="CURRENCY_CODE" default="%InstallationCurrency" required="true"/>

Mnemonic	Valid Values	Description	Examples
			</schema>
	"%InstallationCountry"	You can default the country from installation record.	<schema> <country mapField="COUNTRY" default="%InstallationCountry" required="true"/> </schema>
	"%InstallationLanguage"	You can default the language from the installation record.	<schema> <language mapField="LANGUAGE" default="%InstallationLanguage" required="true"/> </schema>
	"%Constant( )"	You can default an element value using a schema constant.	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'. <language mapField="CUSTOMER_LANG" default="%Constant(customerLanguage)" required="true"/>
	"%Context( )"	You can default a value contained in a context variable. **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.	An example of a context variable used as a default value: <source mapField="PER_ID" default="%Context(source)" required="true"/> **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.
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
is=	"value"	Use this attribute to reference a value directly.	`<tdTypeCd mapField="CHAR_VAL_FK1"> <row mapChild="F1_EXT_LOOKUP_VAL_CHAR"> <CHAR_TYPE_CD is="CM-TD-TYPE"/> </row> </tdTypeCd>`
	"%Constant( )"	You can configure a flattened element using a schema constant. During a service interaction the value of the schema constant will be used to	An example of a schema constant used in flattening syntax where the Java Value Name of the Lookup field value is 'cmRate'. `<unitRate mapField="CHAR_VAL"`

Mnemonic	Valid Values	Description	Examples
		identify the flattened element in its child row.	dataType="number"> <row mapChild= "F1_EXT_LOOKUP_VAL_CHAR"> <CHAR_TYPE is ="%Constant(cmRate)"/> </row> </unitRate>
	"%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.	An example with a relative list instance - where the **first** instance of the row is returned. <schema> <mainPhone mapField="PHONE"> <row mapChild="CI_PER_PHONE"> <PHONE_TYPE_CD is="%Constant(mainPhoneType)"/ > <SEQ_NUMis="%1"/> </row> </mainPhone> </schema>

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 an 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.
			`<schema>`
			`<price mapField="CHAR_VAL"`
			`dataType="number">`
			`<row`
			`mapChild="CI_SA_CHAR">`
			`<CHAR_TYPE is="PRICE"/`
			`>`
			`<EFFDT`
			`is="%effectiveDate"/>`
			`</row>`
			`</price>`
			`</schema>`

	"%effectiveDate( )"	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.
			`<schema>`
			`<price mapField="CHAR_VAL"`
			`dataType="number" required="true">`
			`<row mapChild="CI_SA_CHAR">`
			`<CHAR_TYPE is="PRICE"/`
			`>`
			`<EFFDT`
			`is="%effectiveDate(priceEdate)" /`
			`>`
			`</row>`
			`</price>`
			`<priceEdate mapField="EFFDT"`
			`rowRef="price"/>`
			`</schema>`

	"%effectiveDateTime"	Use this configuration to indicate that current date /time should be used for processing. Any value added or updated using this schema will be for the current date / time.	`<schema>`
			`<price mapField="CHAR_VAL"`
			`dataType="number">`
			`<row mapChild="RATE_CHAR">`
			`<CHAR_TYPE is="PRICE"/`
			`>`
			`<EFFDT`
			`is="%effectiveDateTime" /`
			`>`
			`</row>`
			`</price>`
			`</schema>`

	"%effectiveDateTime( )"	Use this configuration to indicate that the date / time to use the value of another element.
		NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.
			`<schema>`
			`<price mapField="CHAR_VAL"`
			`dataType="number">`
			`<row mapChild="RATE_CHAR">`
			`<CHAR_TYPE is="PRICE"/`
			`>`
			`<EFFDTTM`
			`is="%effectiveDateTime(priceEdatetime)"`
			`>`
			`</row>`
			`</price>`
			`<priceEdatetime mapField="EFFDTTM"`
			`rowRef="price"/>`
			`</schema>`

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

The following table highlights the supported include statements.

Mnemonic	Description	Examples

<includeBO name=" "/>	Including a business object schema into another schema is allowed.
	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="C1-Person"/>`
		`</cust>`
		`  <spouse type="group">`
		`<includeBO name="C1-Person"/>`
		`</spouse>`
		`</schema>`

<includeBS name=" "/>	Used to include a business service schema.	`<schema>`
		`<includeBS name="F1-ReadMOLog"/>`
		`</schema>`

<includeDA name=" "/>	Used to include a data area schema.	`<schema>`
		`<includeDA`
		`  name="F1CommonSchemaFieldData"/`
		`>`
		`</schema>`

<includeMP name=" "/>	Used to include a UI map schema.	`<schema>`
		`<includeMP`
		`  name="F1-DisplayRecordActions"/`
		`>`
		`</schema>`

<includeSS name=" "/>	Used to include a service script schema.	`<schema>`
		`<includeSS name="F1-ActShowZn"/>`
		`</schema>`

Compatibility Attributes

These attributes were added as part of upgrades from pervious 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 and input maps have functionality that requires 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:
  - **F1-AttachmentMain**. This is the main section 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.

- **F1-AttachmentActions**. This record actions map contains the standard actions, Edit and Delete, plus custom actions used only by attachments, View and Upload.
- **F1-AttachmentIDFrag**. This record information 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:
- **F1-AttchDtlU**. This display map 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:
- **F1-AttchPre**. This pre-processing 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 post-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:
- **F1-AttchPost**. This post-processing 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

NOTE: Throughout this topic the term "field" to refer to both the generic concept of displaying and capturing data in a 'field' as well as referring to the meta-data object supplied in the product to define Fields. When referring to the latter, the term "MD Field" (meta-data Field) is used.

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.

Syntax	Description	Examples
<uiHint:title mdField=" "/>	Displays the label of a referenced MD field as the title.	`<schema xmlns:uiHint="http://oracle.com/ouafUIHints"><uiHint:title mdField="STATUS_RSN_LBL"/> ... </schema>`
<uiHint:title text=" "/>	Displays the indicated text as the title. (Do not use this mechanism when multiple languages are supported.	`<schema xmlns:uiHint="http://oracle.com/ouafUIHints"><uiHint:title text="Status Reason"/> ... </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.

Syntax	Supporting Attributes	Description
<uiHint:startSection .../>	sectionColumn="left | right | fullWidth | float"	The default is that the section will be the full width in display maps. To override that setting, specify if you want a half-width section to appear in either the left (**left**) or right (**right**) column or to float (**float**). Sections that are marked as 'float' will display half-width and be aligned according to whether prior sections are displayed or conditionally hidden. For

Syntax	Supporting Attributes	Description
		example, if a left-aligned section is followed by a floating section, the floating section will appear in the right column if the left section is populated but will display in the left column if the left section is hidden / collapsed.
	editColumn="left | right | fullWidth | float"	By default a section appears as full width in maintenance maps. To override that setting, specify if you want a half-width section to appear in either the left (**left**) or right (**right**) column or to float (**float**). The behavior is the analogous to the **sectionColumn** behavior.
	sectionOpen="false"	By default a section is open on initial display. Specify this attribute to initially display the section as closed (collapsed).
	mdField=" "	Specify the name of a MD field whose label should be used as the section heading.
	label=" "	Specify the explicit text to use as the section heading.
	visibleOn="displayMap | inputMap"	By default a section appears on both the display and the input maps. Use this attribute to limit the display of the section to either the display map (**displayMap**) or input map (**inputMap**).

The syntax for the end section attribute is **<uiHint:endSection/>**

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    <uiHint:startSection label="Main" sectionColumn="left"/>
    ...
    <uiHint:endSection/>
</schema>
```

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    <uiHint:startSection mdField="F1-ADD-
INFO" sectionColumn="fullWidth" editColumn="float" sectionOpen="false" visibleOn="displayMap"/>
    ...
    <uiHint:endSection/>
</schema>
```

NOTE: The sectionColumn, editColumn and sectionOpen attributes are available for group and list nodes as well.

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.

Syntax	Supporting Attributes	Description
<uiHint:includeMap .../>	map=" "	Specify the name of the map.
	visibleOn="displayMap | inputMap"	By default the details from the map fragment appear on both the display and the input

Syntax	Supporting Attributes	Description
		maps. Use this attribute to limit the display of the section.

Example:

```
<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">
/* <![CDATA[ */
//
//javascript
//
/* ]]> */
</script>
```

Flush the cache: 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 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.

Syntax	Description
uiHint:select="ss: "	Specify the name of the service script after the colon.
uiHint:select="bs: "	Specify the name of the business service after the colon.
uiHint:select="table: "	Specify the name of the table after the colon.

When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.

Syntax	Values	Description
uiHint:selectIn=" "	serviceXPath:element	Used to pass the value of another element into the service (mapping to the service's XPath).
	serviceXPath:'Literal'	Used to pass a constant or literal to the service (mapping to the service's XPath).

Syntax	Values	Description
uiHint:selectOut="valuePath: ; descPath: "	See examples below.	Used to indicate which element in the service's output holds the values and which one holds the descriptions.

Examples:

```
<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" fkRef="F1-
OMTYP" uiHint:select="table:F1_OUTMSG_TYPE"/>
</schema>
```

Conditionally Hide Elements

The **displayNone** attribute is used to suppress elements on the map based on conditions.

Syntax	Values	Description
uiHint:displayNone=	"'XPath','value','!=' | '='"	Used to conditionally hide this element based on the value of another element (referenced using its XPath). Enter a value of ' ' to interrogate a blank value. By default the operator is **'='**. This may be overridden using **'!='**.
	"function name, **true** | **false**"	Used to indicate a JavaScript function, which must return a Boolean.

WARNING:
Embedded spaces are not supported within the comma separated string values of this attribute.

This setting 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 two elements (currency reference and lookup) that will be hidden or displayed based on the value of the data type element. 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 subsequent elements 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="currencyRef;lookup; "/>
     <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'),true"/>
```

Conditionally Protect Elements

The protect attribute is used to protect elements on the map based on other factors.

Syntax	Values	Description
uiHint:protect=	"'XPath','value','!=' | '='"	Used to conditionally protect this element based on the value of another element (referenced using its XPath). Enter a value of ' ' to interrogate a blank value. By default the operator is **'='**. This may be overridden using **'!='**.
	"function name, **true** | **false**"	Used to indicate a JavaScript function, which must return a Boolean.
	"'**action**','A' | 'C','!=' | '='"	Use the **'action'** setting to protect the element based on the current action. For example, certain elements may only be specified when adding a record. Any subsequent changes to the record should protect the element from being changed. When using this option, the valid values for the *'value'* are **A** (add) and **C** (change).

WARNING:
Embedded spaces are not supported within the comma separated string values of this attribute.

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 statistics category when the action is 'C'.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
 ...
 <statisticsCategory dataType="lookup" mapField="STAT_CATEGORY_FLG"
    lookup="STAT_CATEGORY_FLG" uiHint:protect="'action','C','='"/>
 ...
</schema>
```

Trigger Dependent Behavior

The dependents attribute is used to trigger behavior on a child element when a parent element is changed.

Syntax	Values
uiHint:dependents=" "	A list of one or more dependent elements separated by semicolons.

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:dependents="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.

Syntax	Values
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.

Syntax
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.

Syntax
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>
```

FASTPATH: Refer to FK Reference Formatting in the UI Map Attributes section for more information on each FK reference setting.

Suppress Automatic Number Formatting

By default numeric fields (**dataType="number"**) are formatted as numbers. An attribute is provided to instead apply alphanumeric formatting.

Note: If **dataType** is not specified explicitly, it is be derived from **mdField** or **mapField**.

Syntax

uiHint:alphaFormat="true|false"

By default, its value is **false** (and therefore can be left out altogether).

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
 ...
 <numberCount mdField="" dataType="number" uiHint:alphaFormat="true"/>
    ...
</schema>
```

Auto Capitalize the Input Data

The uiHint provides syntax to automatically capitalize input data.

Syntax

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 attribute is ignored if **uiHint:textArea="true"** is configured.

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.

Schema Designer

The Schema Designer is a user-friendly interface for performing the following common schema editing tasks:

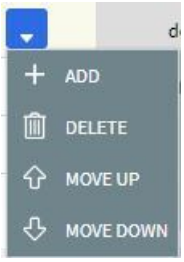
- Displaying existing schemas.
- Creating schema elements.
- Moving elements within a schema.
- Adding attribute values.

The designer provides two view modes:

- **Text** mode shows the schema elements and their attributes written in the proper syntax and allows for direct text entry.
- **Tree** mode is a view showing the elements in a tree format with many of the key attributes of each element in tabular form. While in this mode, the **Node Display** controls allow the user to choose between displaying the elements by their internal identifiers or their associated screen labels while viewing or editing the schema.

NOTE: Schemas that are not owned by the current installation owner are protected in both view modes.

The following sections provide more information about functionality available in the Tree mode.

- If the schema definition refers to another schema using an 'include' statement, a triangle is visible to the left of the tree area. One can expand that schema by clicking the triangle.
- Detailed information about an element's definition can be displayed and if applicable, updated from the**Tree** view by clicking the edit icon for that element, which appears on the right. Note that the element attributes are only editable if the element is defined in this schema. When viewing the attributes of an element that is part of an 'include' of another schema, the attributes are display only.
- Context-sensitive embedded help is provided for fields and controls in the edit pane by clicking the Help icon
- There is a menu dropdown icon visible to the left of the tree for any element that is defined in the current schema being viewed. It is not visible for elements included from another schema. New elements may be added in the **Tree** mode. Clicking that dropdown displays options to Add, Delete or Move an element. 

Note that not all options are visible, based on what is allowed for the element adjacent to the menu. For example, on the main "schema" node, only the Add option is visible as deleting or moving the "schema" node is not applicable.

When adding a new element, you are prompted first for the element position, which is either a sibling node to the current element or a child node of that element. For either option, the new element is added below the one adjacent to where the menu is clicked. You are then prompted for the element type.

The following lists the possible element types. Most are self explanatory and represent standard schema options.

- Characteristic List. This is a special type of Flattened List element that is used to map list elements that include 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.
- 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.
- 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).

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.

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 items that are suppressed are not visible in the Menu Item Search results.

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.

Accessibility Considerations

This section provides information about topics related to accessibility that should be considered when extending the system via configuration tools.

Color Contrast

The HTML color **red** (RGB code #FF0000) when used for text on a white background does not comply with the accessibility color contrast guidelines. When it is desired to use red for text for emphasis in a zone, map or in a text string, the product recommends the shade of red with the RGB code #E0292F.

Note that in addition, the product provides a special CSS class **textColorRedOnWhite** which may be used when HTML references are used for applying the red color. For example, UI map, zone help text and scripting.

Referencing URIs

There are some configuration objects that require a reference to a URI, including file path URIs. The following sections highlight some functionality supported by the product with respect to defining / accessing URIs.

NOTE: In order for the functionality described below to occur, specific APIs must be used by the underlying code related to the fields that capture or process the file path or URL information. If you find that there is a URI-related field that does not provide the functionality described here, please contact customer support.

NOTE: For schema elements in a business object that reference a Field with the **URI** data type or define the element with the **URI** data type configured in the schema will automatically use the appropriate API that validates the value that may reference a substitution variable or must be checked against the whitelist if applicable.

Validation Against a Whitelist

Based on a property setting, your implementation may be configured to define a whitelist of URIs. If this setting is enabled, the system will issue an error if the URI is not defined in the whitelist. Consult your system administrator to verify if this setting is enabled or not for your implementation.

FASTPATH: Refer to URI Substitution for information about how defining substitution tokens for URIs using the properties file technique automatically adds the defined URI to the whitelist.

URI Substitution

The system supports the ability to define substitution variables (sometimes referred to as tokens) for both URL values and file path values that reference native file storage locations. For URLs, the system supports defining variables in a Substitution Variable properties file. For native file storage paths, there are two options: the Substitution Variable properties file or via a File Storage configuration extendable lookup. More details on both these options are found below.

- Substitution variables properties file. A substitution variable for all or part of the URI definition may be configured in a properties file. This allows the system administrators to define the proper URI locations in a properties file whereas the configuration users only need to know the variable name. For example, when defining a location for an extract file in an extract batch job, instead of typing a file path of **h:\oracle\serverName\1.0.0.0\batch\extract**, the batch user can enter **@FILE_EXTRACT@**, assuming there is an entry in the substitution variables file with a name of **FILE_EXTRACT**, and a value of **h:\oracle\serverName\1.0.0.0\batch\extract**. Another example is that the batch user could enter **@BATCH_FILES@\extract**, assuming that the URI variable for **BATCH_FILES** is defined as **h:\oracle\serverName\1.0.0.0\batch**.

  NOTE: The product automatically populates the value of SPLOUTPUT in the properties file so that this may be used in URI configuration. In addition, the product may supply some pre-defined variable names for other common references. As part of this, the 'advanced' menu in the system installation steps may prompt for installers to define the values of these pre-defined variables, if desired. Installations may opt to define additional substitution variables for various URI references. Refer to the *System Administration Guide* for more information.

- File Storage extendable lookup. Specifically for file paths that reference the native file system, the system also supports the ability to define a path using the **File Storage Configuration** extendable lookup. The following points highlight the steps to take for this option.
  - Navigate to Extendable Lookup and search for the **File Storage Configuration** lookup.
  - Click **Add** to create a new entry. Define a lookup value name. This will be used when configuring a file path that uses this value. Choose the File Adapter value of **Native File Storage**. Enter the desired file path value. The values defined here may in turn refer to values defined in the Substitution Variables properties file. For example, the file path can reference @SPLOUTPUT@.
  - To reference this value in system configuration, use the syntax **file-storage://XXXX**, where XXXX is the extendable lookup value. Using the same example from above, if you define an extendable lookup value of CM-FileExtract with a file path of **h:\oracle\serverName\1.0.0.0\batch\extract**, then when configuring the extract file path for an extract

batch job, enter **file-storage://CM-FileExtract**. Another example is that if the extendable lookup value's file path is configured as **h:\oracle\serverName\1.0.0.0\batch**, then the user configuring the file path on the batch extract can enter **file-storage://CM-FileExtract/extract**

When should you define substitution variables in the properties file and when you define them in the extendable lookup? The following points highlight differences between the options that may help this decision.

- Substitution variables for URLs are only supported via the properties file.
- The properties file can typically only be modified by a system administrator. If there are values that are set at installation time and don't change, then defining the values in the properties file may be beneficial.
- When defining additional environments, such as test environments or production, the values in the extendable lookup may be copied using CMA. Ideally the values are defined such that they are the same between various regions.

External File Storage

The system supports using Oracle Cloud Object Storage for cloud customers for managing files and can be configured to read files from or write files to this external location. Refer to Oracle Cloud Object Storage documentation for more information about obtaining an account and defining appropriate file locations or "buckets".

Once your cloud storage information is defined, the following points highlight the configuration steps required in the application.

- Define a signature key ring. When the system tries to communicate with cloud object storage, it must provide a signature key so that cloud storage can confirm that the request is from a trusted source.
  - Navigate to Key Ring in add mode and select the **Signature Key Ring** business object.
  - Define a key ring code, which will be used in the File Storage configuration (below) along with a description.
  - Once the key ring is added, click **Generate Key** to generate a private / public key pair.
  - Click View Public Key to launch a pop-up that displays the public key, allowing the user to copy the key.
- At this point, the user should navigate to cloud object storage and register the public key. Once this is done, the key ring is now ready to be configured in file storage configuration.
- Use a File Storage extendable lookup to define the location and connection information so that you may reference this location in system configuration. The following points highlight the steps to take for this option.
  - Navigate to Extendable Lookup and search for the **File Storage Configuration** lookup.
  - Click **Add** to create a new entry. Define a lookup value name. Note that the lookup value should not have a slash or backslash in its name. This will be used when configuring a file path that uses this value. Choose the File Adapter value of **Oracle Cloud Object Storage**. Provide the following information that identifies the cloud storage options: User, Tenancy, Compartment, Namespace and Region. For the Key Ring, choose the value defined above.
- To reference this value in system configuration, use the syntax **file-storage://XXXX/...**, where XXXX is the extendable lookup value and any additional path information that is appropriate. For example, if you define an extendable lookup value of **CM-CloudStorage** and you have a bucket defined in Cloud Storage for processUpload, when configuring the file path for an upload batch job, enter **file-storage://CM-CloudStorage/processUpload**.
- You may optionally use a Bucket Name Prefix on the file storage configuration to support defining different Bucket Names for different environments. For example, imagine you want to have a separate "processUpload" bucket for a development region, a test region and a production region. In the Cloud Storage configuration you can define separate buckets for "DEV_processUpload", "TEST_processUpload" and "PROD_processUpload". If you needed to define the full bucket on each file path reference on various batch jobs, migrating the batch job configuration from one environment to another would require a manual step to change all the bucket references. Instead, the system supports defining a Bucket Name Prefix, which if populated will be prepended to bucket reference before connecting to Cloud Storage. This means that the prefix is defined once and all the references throughout the system can keep the common part of the

Bucket name (such as "processUpload"). Then only the File Storage configuration needs to differ between the different regions.

NOTE: The initial key pair is generated in the **Active** state. Once the key ring is defined on a File Storage lookup, it is recommended to practice key rotation and generate new keys periodically. To do this, use the **Generate Key** button on the Key Ring. New key pairs are generated in the **Inactive** state. The user should register the public key of the new key pair with cloud storage. Once that is done, the new key should be activated. The system generates the signature for connecting to cloud storage using the private key of the **Active** key pair for the key ring.

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.

Also note that the system provides a business service **F1-FlushAllService**, which is equivalent to the flushAll command (except that it does not restart your browser session). This may be useful for any processes that are built orchestrate updates to configuration data where you want to ensure that the cache is immediately flushed. This business service is also exposed as a REST web service.

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**. This is often referred to as the "L2 cache". 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 are any batch jobs that update any tables that are marked as **Cached for Batch**, you must run that batch job using a special thread pool that is configured with the caching disabled. Contact your system administrator to confirm the thread pool that is configured this way.

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 section, 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 (note that depending on the browser you use, the steps may vary slightly):

- 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. **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	Parameter	Results	Comments
size( )	List element	Number of elements in the list.	
isEmpty( )	List element	Returns true if the list is empty.	
sum( )	List element of type 'number'	Returns the sum of the numbers in the list.	
avg( )	List element of type 'number'	Returns the average of the numbers in the list.	
	One or more numbers separated by commas	Returns the average of the number arguments.	
max( )	List element	Returns the largest value in the list.	
	One or more comparable elements.	Returns the largest value of the number arguments.	
min( )	List element	Returns the smallest value in the list.	
	One or more comparable elements.	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, as it us double float based functions.
asin( )	Number	Returns the arc sine of the number radians.	The result will lose precision, as it us double float based functions.
atan( )	Number	Returns the arc tangent of the number radians.	The result will lose precision, as it us double float based functions.

Function	Parameter	Results	Comments
cos( )	Radian	Returns the cosine of the radian angle input.	The result will lose precision, as it us double float based functions.
exp( )	Number	Raises e to the number power.	The result will lose precision, as it us double float based functions.
log10( )	Number	Takes the log, base 10, of the number.	The result will lose precision, as it us double float based functions.
log( )	Number	Takes the natural log (base e) of the number.	The result will lose precision, as it us double float based functions.
sin( )	Radian	Returns the sine of the radian angle input.	The result will lose precision, as it us double float based functions.
sqrt( )	Number	Returns the square root of the number.	The result will lose precision, as it us double float based functions.
tan( )	Radian	Returns the tangent of the radian angle input.	The result will lose precision, as it us double float 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.	
	Number, Scale	The default rounding mode of “round half up” is applied.	
	Number, Scale, Mode	The mode must be set to one of the following: • **“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 explorer.

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 [ ]	This function returns the value **true** if any of the entries in list satisfies the expression.	The following returns **true** if any entry in the Balance list is greater than 0. `any [ i in list/Balance | i > 0 ]`
all [ ]	This function returns the value **true** if all of the entries in the list satisfy the expression.	The following returns **true** if all phone numbers are populated. `all [ i in list/phoneNumber | i != ' ' ]`
collect [ ]	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 new list with the tax rate applied to each amount. `collect [ i in list/amount | i * taxRate ]`
select [ ]	This function returns a list of all the values of the original list that satisfy the Boolean expression.	The following returns a new list with only the amounts that are negative numbers. `select [ i in list/amount | i < 0 ]`

Function	Description	Examples
reject [ ]	This function returns a list of all the values of the original list that do not satisfy the Boolean expression.	The following returns a new list with only the amounts that are not negative numbers. `reject [ i in list/amount | i < 0 ]`

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.

The following are only applicable in a non-Cloud environment:

- **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.

The following are only applicable in a non-Cloud environment:

- **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 page into which a user is taken when they drill down to the related object on an entry.
- The message associated with the To Do that appears on various pages. 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 entry sort keys. Note that the processes that create To Do entries are responsible for populating the sort key values.
- 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 entry. Note that this value may be overridden by a **Calculate Priority** algorithm.
- An indication of whether a To Do of that type may be created manually by a user.

- The algorithms used to perform specific business rules for To Do entries of this type.
- The characteristics applicable to the To Do.

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 may be many ways to default an entry's role. However, the base package processes and algorithms use the following mechanisms to determine the default:

- 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 particular account management group and that account management group includes configuration for a special role for certain To Do types. Refer to Linking Additional Information to a To Do Entry 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.

Supervisors for a Role

Most organizations have the notion of a supervisor who is responsible for all entries assigned to a given role. A user that is considered a supervisor should be a valid user in a role. In addition, application security is used to provide supervisors access to additional capabilities, namely the ability to review To Do entries assigned to other users and the ability to assign To Do entries to other users.

The pages listed in To Do Supervisor Functions are meant for supervisors only. So security for these pages should be restricted to those users that are considered supervisors. The To Do Management portal and the To Do Search page are available to all users. There is a special access mode for Supervisors to allow those users the ability to assign To Do entries to other 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. To Do entries may also be suppressed by the same algorithms that are responsible for performing To Do pre-creation logic. Refer to Linking Additional Information To A To Do Entry for more information.

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 based on message number. 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 assigned as follows:

- If one or more **Calculate Priority** plug-ins are defined on the To Do entry's type, the system calls them to determine the entry's priority. They are called initially when a To Do entry is created and each time it gets updated. 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. The system also provides a batch process (**F1-TDCLP**) that calls the calculate priority algorithms "at will" for non-closed To Do entries. This is useful when the priority should be 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. When the priority is determined by one of the algorithms, a log entry is created indicating that the priority was calculated.
- 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. When a user overrides the priority, a log entry is created indicating that the priority was overridden. 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.

Monitoring A To Do Entry

A To Do type may reference a To Do monitor algorithm. The algorithms are executed by the To Do monitor background process (F1-TDMON). The process executes the monitor algorithms for any non-complete To Do entry whose To Do type references at least one monitor algorithm.

Monitor algorithms may be useful if there are business requirements where a To Do could be automatically completed based on criteria that the algorithm can review. For example, an algorithm may be used to detect if the situation that caused the To Do Entry to be generated has been remedied in the meantime, allowing for the To Do Entry to be completed.

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
- The To Do entry's priority was updated as a result of a calculate priority algorithm.

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 that the Plug-in Driven batch program cannot be used for this purpose. This program assumes that any To Do Type that references it as a creation process is using the link for the Object-Specific Error Condition use case. If you want to use the plug-in driven batch process to create To Do entries based on a given condition, then you need to follow the pattern described in To Dos Created by Background Processes for Specific Conditions.

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, drill key and message category / message number. 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:

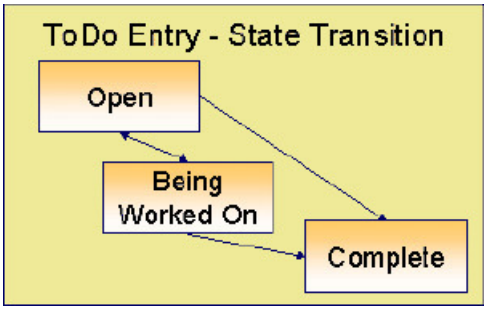
When a user adds a manual To Do entry, it is created with a drill key value equal to the To Do entry's ID. When the user clicks the Go To button next to the message in the To Do list, the system uses the drill down application service (defined on the main tab) and the drill key to display the associated 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 theBeing Worked Onstate.** 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. This may also be referred to as "reopening" or "unassigning" a To Do entry.

When an entry becomes **Complete**, it is no longer visible in various To Do 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.

- The To Do monitor process may be used to evaluate whether or not the To Do entry can be closed automatically. Refer to Monitoring a To Do Entry for more information.

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. 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.

When creating an automatic To Do entry, the program that generates the To Do may link related data to the To Do using characteristics. In addition, the system **To Do Pre-creation** algorithms can perform processing to populate additional data prior to creating the To Do, based on information that the algorithm can collect. Algorithms of this type are most commonly used for:

- 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. Other edge applications provide a similar algorithm for their relevant entities. Note, before you can set up these algorithms, 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.

The system provides the ability to configure**To Do Pre-creation** algorithms on the To Do Type or the Installation record. In general, Installation pre-creation algorithms would be used for adding context or performing overrides that apply to many To Do Types and To Do Type pre-creation algorithms would be used for linking context data that is specific to that type.

If at least one To Do pre-creation algorithm is plugged in on the To Do Type, only those algorithms are executed when creating the To Do entry and the Installation Option algorithms are ignored. However, the system provides a To Do Type pre-creation algorithm that executes the Installation option To Do pre-creation algorithms. This allows you to control whether the installation algorithms should also be executed and if so, when.

In addition to linking information to the To Do entry, both the To Do Type and Installation option pre-creation algorithms may be used to indicate that the To Do Entry should be suppressed. The expectation is that algorithms that determine if an entry should not be created will be configured on the To Do Type, as this allows for more granular conditions to be checked for a given To Do entry.

If your To Do entries reference characteristics that are related to your global context data, you may want to configure an alert algorithm to display an alert if a related entry is **Open** or **Being Worked On**. Refer to Installation Options for more information.

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.

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.

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

There are a number of installation options that may be configured to govern various aspects of To Do processing.

- A **To Do Information** algorithm may be plugged in on the Installation record. Refer to To Do Entries My be Formatted By An Algorithm for more information.
- 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. Refer to Linking Additional Information To A To Do Entry for more information.
- A **Next To Do Assignment** algorithm must be plugged into the Installation record if your organization opts to use the next assignment feature supported by the Current To Do dashboard zone. The algorithm is responsible for determining 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. Refer to the Current To Do zone for more information.
- 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 an 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.

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-TDSK** has the ability to assign To Do entries to users that have the proper skills to work on them. This plug-in is 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.

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** > **General** > **To Do Role** 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.

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 attributes that may be used to sort To Do entries.

The To Do Information zone on the To Do Management portal provides the ability to search for To Do entries using sort keys and includes up to five sort key values as columns in the results, which may be used for sorting as required.

The To Do list page includes an **Extra Information** option to allow To Do entries to be sorted by a selected sort key. Note the default sort key for this type of sorting is also defined on To Do type.

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 various pages.

Use as Default indicates the default sort key that is initially used when a user opens the To Do List page. 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 on the To Do List page.

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	Algorithms of this type may be used to calculate a To Do entry's priority. Refer to The Priority of a To Do Entry for more information on when this plug-in is called. 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. If more than one algorithm is plugged-in the system calls them one by one until the first to return a calculated priority.
External Routing	Optional	Algorithms of this type may be used to route a To Do entry to an external system. 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. 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.
To Do Information	Optional	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. Plug an algorithm into this spot to override the "To Do information" algorithm on installation options or the system default "To Do information" if no such algorithm is defined on installation options.
To Do Monitor	Optional	Algorithms of this type are executed by the To Do Monitor background process and may be used to periodically review a To Do entry and perform actions, if needed.

System Event	Optional / Required	Description
		Refer to Monitoring a To Do Entry for more information.
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.
To Do Pre-Creation	Optional	Algorithms of this type are called when a To Do entry is being added. They are typically used to set up additional information for the To Do, such as characteristics. They may also set a flag to indicate that the To Do entry should be suppressed. To Do Type pre-creation algorithms override the Installation level pre-creation algorithms.

Implementing The To Do Entries

To enable the To Do entries visible in the To Do Type page, 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:

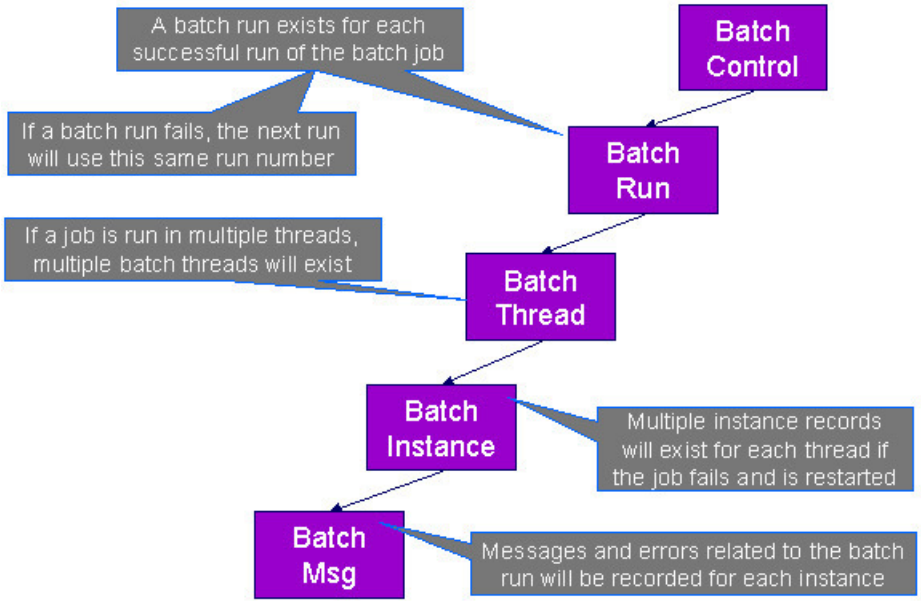
- Processing To Do Entries
- Monitor processes that select records in a given state to progress them to their next state in their lifecycle
- Processes that purge data
- Processes that extract data
- And many more...

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, a billing process may use 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:
  - Both the user submitting the job and the user ID recorded on the batch submission should have access to the application service for the batch control that secures execution.
  - Any batch process that stamps a user ID on a record it creates or updates uses this user ID in applicable processing.
  - 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**.

Parameter Name	Description	Additional Comments
emailMode	When the batch job is submitted with an associated email address, the default logic is to send an email when the job completes regardless of success or failure. Use this parameter to limit the email based on the status of the job when it ends.	Valid Values • **ERROR** — send an email only when the job ends in Error status. • **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. 9999999999), 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. 9999999999), 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 com.splwg.base.api.batch.BatchIdRangeOverride.
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.

Indicating a File Path or File Name

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.

For the file path parameter, refer to Referencing URIs for several points related to populating the file path and the supported syntax.

For the file name parameter, there are extract processes that support defining a file name that ends in ".zip" or ".gz", which will automatically result in writing the file with zip or gzip compression (based on the suffix). Similarly, there are upload processes that support decompression of files using the appropriate method based on the file name ending in ".zip" or ".gz".

If the extract or upload process supports compression / decompression, it should be indicated accordingly in the description of the File Name parameter. If a batch control doesn't explicitly indicate that it supports compression or decompression in its description, the assumption should be that it is not supported.

Manifest File

Batch processes that create files, for example data export related batch processes, may create one or more files in each processing thread. External applications that consume these files need to process them only when the entire batch process is successfully completed. To automate the consumption of such files, the external application may leverage a function to be triggered upon the creation of a manifest file to process all the files referenced by the manifest. The manifest file includes details such as content hash value, file size information etc that assist in confirming that file content is completed and has not been tampered with prior to processing the files.

The following sections provide additional information about manifest files.

The Manifest Option Batch Parameter

Manifest file creation is supported on certain base product extract related batch controls only. These batch controls include the **Manifest Option** batch parameter that controls whether a manifest file should be created or not upon successful completion of the batch process. You may use the Batch Control Query portal to find all batch controls supporting this option.

Manifest File Information

Manifest file content is provided is JSON format and includes the following information:

- The version of the manifest details structure.
- The batch control code, run number and rerun number identifying the batch run producing the files
- Total number of files created by the batch run.
- Total number of records processed by the batch run.
- The manifest creation timestamp in ISO format.
- A list of files created by the batch run that includes the following details for each file: the file’s name, its size in bytes and the SHA-256 hash value of the file’s content.

The manifest file name is constructed as follows:

```
[batch control]_[batch run number]_[batch rerun number]_manifest.json
```

The following is an example of the manifest file structure.

```
{
  "version": 1,
  "batchCode": "HD-PDE",
  "batchRunNumber": 106,
  "batchRerunNumber": 0,
  "totalFiles": 3,
  "recordsProcessed": 10,
  "timeStamp": "2020-05-02T16:55:43.000353Z",
  "fileDetails": [
    {
      "fileName": "SOME_FILE_NAME_001.json.gz",
      "fileSize": 1543,
      "hashAlgorithm": "SHA-256",
      "hashValue": "ff7d4e466f93cadc4a18ab009080916baff631e03b728e2f2684c4a4f92ccbe1"
    },
    {
      "fileName": "SOME_FILE_NAME_002.json.gz",
      "fileSize": 1182,
      "hashAlgorithm": "SHA-256",
      "hashValue": "78869e7ce971f28a70f5b35693a05726ba66109fecf492edb60291a62ae1ba9f"
    },
```

```
    {
      "fileName": "SOME_FILE_NAME_003.json.gz",
      "fileSize": 1182,
      "hashAlgorithm": "SHA-256",
      "hashValue": "78869e7ce971f28a70f5b35693a05726ba66109fecf492edb60291a62ae1ba9f"
    }
  ]
}{
```

Verifying The Manifest File

The **Verify Manifest File** web service may be used by an external system to validate that the manifest file it has received for a specific batch run is not corrupted or tampered with. The service returns the manifest file name and the hash value of its content for the requested batch run allowing the caller to compare their file details with the returned values.

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.

Error Post-Processing Logic

The product supports executing one or more algorithms when a batch process encounters an error that causes execution to stop. This allows for some special processing to occur to handle the failure of the batch job. Algorithms for this plug-in spot receive the batch control, batch run number, batch processing business date, number of threads and the list of the ad hoc parameters of the batch job.

> **NOTE:** This plug-in spot is available for all Java based batch programs. For programs of type **Java (converted)**, the individual batch programs need to explicitly support this plug-in spot. Unless otherwise noted, assume that a **Java (converted)** program does not support it.

The following are some examples of functionality that may be executed when a batch job fails:

- Another object or record that is monitoring the batch job may have its status updated to reflect the batch status.

- An outbound message service may be invoked to perform a task related to the failure.

Note that the units of work for all threads are committed prior to executing the error post-processing logic.

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, batch run number, batch processing business date, number of threads and the list of the ad hoc parameters of the batch job.

> **NOTE:** This plug-in spot is available for all Java based batch programs. For programs of type **Java (converted)**, the individual batch programs need to explicitly support this plug-in spot. Unless otherwise noted, assume that a **Java (converted)** program does not support it.

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. The product also provides a batch control template (**F1-MNTPL**) that should be used as a starting point. 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 given 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.
restrictToBOStatus	Restrict by Status	This parameter may be used to limit the processing to records that are in the indicated status.
restrictToDate	Restrict By Date	Enter a valid date on the record's primary table to limit processing to records that are have a value in this field that is on or before the batch business date. Optionally, enter "+" or "-" followed by a number to shift the comparison date to the batch business date plus or minus the given number. For example, entering REVIEW_DT -3 will retrieve all records whose review date is on or before the batch business date minus 3 days.
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 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 do the work that is needed. These are called plug-in driven background processes. There are two major types of plug-in driven batch processes:

- Processes that act on records that are stored in the database in the system. These types of processes require SQL to select the records along with the logic to process the records.
- Processes that import data from a file and store new records in the system as a result. These types of processes require an algorithm that is able to map the data from the file to appropriate new records in the system

The subsequent sections provide more detail about the two types of plug-in driven background processes.

Processing System Records

Processes that retrieve records in the system and do some action on them require an algorithm to select the records to be processed and another algorithm to process the records. The base processes implement standard background process functionality including parallel background process logic and 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 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.

The following sections provide more information about the functionality that is common for both ad-hoc and extract processes. Refer to Extract Record Processing for additional information specific to extract processes.

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 **f1_** as its prefix.

The system supports the following pre-defined bind parameters:

- **:f1_lowID** and **:f1_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.
- **:f1_batchCode** and **:f1_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.
- **:f1_businessDate** - the batch job will populate the input batch business date, if populated otherwise the current date.

NOTE: The system supports both "f1." and "f1_" as a prefix for the bind variables. However, the "f1." prefix will result in an error if the SQL security property setting is turned on. As such, the underscore syntax is recommended. The period syntax remains for backwards compatibility.

For any other custom parameters, the Select Records algorithm may return one or more sets of field name / variable name / value where the variable name matches a bind variable in the SQL. The field name provides information about the length that assists the SQL binding logic to properly substitute the values. Note that the variable name cannot start with **f1.** as its prefix. The batch job will use the value returned by the algorithm to set the bind parameter in the SQL statement. Also note that the logic binds all variables as a string so for data variables, the SQL must include the "TO_DATE" function.

The plug in spot receives a list of the ad hoc parameters for the batch job as name / value pairs. If the list includes parameters whose values are to be used in selecting records, your algorithm may be used to identify the relevant batch parameter passed as input and populate the field name and output bind variable appropriately.

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. Refer to Extract Record Processing for more information on the functionality specific to extract processes.

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.

Configuring a New Process

The following points summarize the steps needed to implement a new background process that acts on records in the system 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 to process each record. If your edge product does not deliver a suitable plug-in, 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. Refer to Extract Record Processing for more information and examples of extract algorithms.

  In either case, if a new processing script is required, 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. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the Select Records and Process Records plug in algorithms.

Extract Record Processing

For an extract batch process, the Process Records algorithms 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). The program will write the data to the file as per the format indicated in the File Format batch parameter. By default the service uses the OUAF format for date and time. To override this and use XSD format, configure the Date Time Format batch parameter to 'XSD".

If an existing schema satisfies the output requirements, it may be used. Otherwise, you should define a data area to indicate the output format of the records as appropriate.

There are two options for designing and configuring an extract process using a plug-in driven background process:

- Create a **Process Records** algorithm type that encapsulates the extract logic. The product provides a base algorithm type that illustrates the technique to follow. Refer to the algorithm type General Process - Sample Process Record Extract (**F1-GENPROCEX**) for more information.
- Create a file integration type that defines the records to be included in the extract and configure a plug-in driven batch control that references that integration type. The following sections provide more information about implementing this type of functionality.

File Integration Type Configuration

Your organization may have many types of extract file integrations. Some can be simple, containing a single record type and possibly a header or footer record. Others can be more complex, involving multiple record types and parent/child relationships between the records.

The framework supplies two objects that work together to define the processing needed to create a specific type of extract:

- The file integration type which describes and classifies the extract configuration.
- The file integration record where the sequence, format and logic to extract the data required for a given record type are captured.

File Integration Type

The file integration type defines the integration classification. Currently, only **Extract** is supported. In addition, the file integration type can be assigned to a category. The category may be used to identify extracts with a common purpose or target such as collection agency referrals. Your edge product may deliver a number of categories or you may define your own.

The system provides a business object for extract file integration types (**F1-ExtractFileIntegrationType**) that is expected to be used for most use cases

File Integration Record

The file integration record is the object that is responsible for defining a record's format and capturing the record details. A separate integration record must be defined for each record type included in the extract.

The record sequence defines the order that records are processed by the system. The parent sequence defines the record type of which this record is a child. Child records may also be referenced as a parent of lower level child records. The base Process Records algorithm provided by the framework processes all child records of a record type before processing the next record type in sequence.

For extracts that are XML format, the system supports creating a grouping node surrounding all the output that belongs to the same entity. File integration records define a record XML node that will be used as the grouping node. Each record type in the list that belongs within the same group would indicate the same value in the record XML node. Refer to the Grouping by Record XML Node section for more details.

> **NOTE:** The system does not support nested record XML nodes. All child records of a record type must define the same record XML node as the parent.

Each file integration record references a data area that defines the extract record's format, allowing the data to be extended. The **Extract Records** algorithms are responsible for populating the data area. The file integration record supports multiple extract processing algorithms so that the custom information can also be populated.

The system provides a business object for extract file integration records (**F1-ExtractFileIntegrationRec**) that is expected to be used for most extract record use cases.

Configuring an Extract Process Using File Integration Type

The following points summarize the additional steps needed to implement a new extract background process that uses a file integration type to define the specific functionality:

- Choose or create data area objects to define the extract record formats.
- Define the logic required to populate each extract record type. Your edge product may deliver suitable plug-ins or you may need to create a plug-in script where the algorithm entity is **File Integration Record - Extract Record**. Note that the Extract Record plug-in receives most of the information defined in the Batch Control - Process Records plug-in.
- Define the algorithm type and algorithm for each newly created script.
- Choose or create a category for the file integration type using lookup **FILE_INT_OBJ_FLG**.
- Create the file integration type and configure the file integration records using the defined data areas and algorithms.
- Create a batch control by duplicating the appropriate base template for an extract background process. Include the File Integration Type parameter on the batch control and reference the appropriate file integration type.
- Plug in a Process Records algorithm that reads the file integration type configuration and processes the record types. The system provides a base file integration record extraction algorithm **(F1-FILEX-PR)** that is expected to be used for most file integration plug-in extract processes.

Extending a File Integration Extract Process

Your edge product may deliver file integration types and batch controls designed for standard extract processes. However, extending the delivered extracts to add custom data is common practice for most implementations. You can customize the output from a delivered extract by:

- Identifying the record types that need additional details
- Creating a data area that defines the additional details and using that to extend the data area configured on the record type
- Creating a File Integration Record — Extract Record plug-in script, algorithm type and algorithm that populates the extra details
- Adding the algorithm to the integration record's extract algorithms collection

In some cases, you may wish to extend the extract by adding additional record types. To do this, add your custom file integration records to the file integration type, following the steps described in Configuring an Extract Process Using File Integration Type.

Grouping by Record XML Node

When configuring an extract process that should produce the output in XML format, your Process Record plug-in may return multiple schemas with information that is all part of the same overall record. For example, the output could include account information and related service agreement details:

```
<account type="group"
    <accountId>1234567890</accountId>
    ...
</account>
<sa type="group"
    <saId>123457665</saId>
    ...
</sa>
<sa type="group"
    <saId>1234588913</saId>
    ...
</sa>
```

In this case, it may be desired to wrap all that information in a grouping XML tag so that all the information for one record is grouped together. Because the Process Record plug-in could be returning header or footer records that are not part of a given selected record's extract information, the batch process doesn't know what schemas returned by the Process Record plug-in belong together logically. The output parameter 'record XML node' in the schema collection should be used to indicate the outer XML node to use to group related information together. For example:

```
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-AccountRecord</schemaName>
<schemaType>F1DA</schemaType>
<data><account><accountId>1234567890</accountId>...</data>
</SchemaInstance>
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-SAInfo</schemaName>
<schemaType>F1DA</schemaType>
<data><sa><saId>123457665</saId>... </data>
</SchemaInstance>
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-SAInfo</schemaName>
<schemaType>F1DA</schemaType>
<data><sa><saId>1234588913</saId>... </data>
</SchemaInstance>
```

All schemas returned from a single call to the Process Record plug-in that have the same record XML node will be grouped in the written output within that XML tag, as per the example below:

```
<root>
<record>
```

```
<account>
    <accountId>1234567890</accountId>
    ...
</account>
<sa>
    <saId>123457665</saId>
    ...
</sa>
<sa>
    <saId>1234588913</saId>
    ...
</sa>
</record>
<record>
<account>
    <accountId>987654320</accountId>
    ...
</account>
<sa>
    ...
</sa>
</record>
</root>
```

Skipping Records

By default, the extract process expects one or more schemas to be returned by each call to the Process Record plug-in. (Ideally when designing any process, the Select Records algorithm only selects records that should be processed and the Process Record algorithm does not need to confirm whether the record should be part of the processing or not.) If however there is a need to check a condition in the Process Record plug-in and it is found that no data needs to be extracted for this record, the process record algorithm should return the 'is skipped' output parameter to **true** so that the program doesn't write an empty row. This parameter is not applicable to the Adhoc Process batch program. In that case, the Process Record can simply not perform any action. No information needs to be returned to the program.

Note that the base process records algorithm for file integration types **(F1-FILEX-PR)** will set the ‘is skipped’ parameter to **true** if the file integration record extract record algorithm returns no data.

Uploading Records

The base product provides a background process to upload data from a file. The batch control Plug-in Driven File Upload Template (**F1-PDUPL**) may be used as a template. The process includes parameters to configure the file path and file name for file to upload along with other parameters to control how to handle missing files and how to rename the file once processed. Refer to the description of the batch control and its parameters for more information.

This background process requires an algorithm plugged into the plug-in spot **File Upload**. This plug-in is called once for a given file. The batch process opens the file and passes to the algorithm the file reader element. The algorithm associated with the batch control is responsible for using provided APIs to read the content of the file and store the data in appropriate table(s) (for example, an appropriate staging table). The base provided process supports uploading multiple files and may be run mutli-threaded to process multiple files. Each file is processed by one call to the File Upload algorithm and supports a single commit for the records uploaded in a given file.

> **NOTE:** Plug-in scripts written to implement this type of algorithm must use the Groovy script engine version as the APIs are not accessible using the XPath scripting language. Refer to Additional Coding Options For Server Scripts for more information.

Note that this step in the upload of data is only one part of a typical upload end-to-end process. This step is sometimes referred to in the product as "Process X". The goal of this step is to get records from a file into database records, with minimal validation and processing. The data should be stored in records that are then processed by a second step (often referred to in the product as the "upload" step, for example "Payment Upload"). The second step, independent of the plug-in driven batch process described here, is responsible for validating the data and should be able to be threaded by individual records and have proper error handling for individual records. Note that depending on the type of data being

uploaded, the product may already supply appropriate tables that the plug-in driven upload process may populate. These could be staging tables, such as payment upload staging. Or they may be records with business objects that have a lifecycle designed to handle uploaded data, for example Business Flag. In such cases, the product will typically supply out-of-the-box background processes to validate and further process the data and finalize the upload. If the data to upload does not already have a base provided staging table, be sure to work with your implementation team to identify an appropriate table to use for the plug-in driven batch upload. In addition, confirm the design for the second step that is responsible for the detailed validation and finalization of the data.

The product supplies sample algorithms to illustrate calling the supplied APIs for processing different types of source data: comma delimited, fixed position and XML. In every case, the sample data supported by the upload uses 'degree day' information to illustrate the process. The system provides sample target records (based on the Fact maintenance object) in order to illustrate the step to store records based on the input data. Note that only sample plug-in scripts have been provided. No algorithm, algorithm type or batch control is configured to use the sample plug-in scripts. To view the scripts, navigate to the Script page, search for a Script Type of **Plug-in Script** and Algorithm Entity of **Batch Control - File Upload** and look for the 'Sample' scripts.

Note that the sample plug-in scripts provided by the product are supplied to illustrate use of the provided APIs. They should not be considered a template for how to implement a plug-in script for a real use case. The following highlight some points to consider when designing a file upload algorithm:

- **Error handling / resolution.** The sample plug-in scripts do some basic error handling related to the data to illustrate error handling. However, any errors found in this step require processing of the whole file to stop. As such, this plug-in should only report errors that are not possible to fix, but where the whole file should be rejected. If there are errors that can be adjusted in the data, then the recommendation is to not check for those errors at this step. Rather, this plug-in should simply populate the appropriate staging tables and let the next step check for validity. As described above, the next step should include the ability to mark individual records in error, allowing for users to fix the data and retry.
- **Target tables.** The sample plug-in scripts use Fact as the target for the resulting insert statements. As mentioned above, the decision of where to store the uploaded data must be carefully considered. There may already be existing tables that are specific to a given use case. If the data being uploaded does not have existing tables to use, review the product to verify what existing tables may be useful, such as Inbound Sync Request or Service Task. Be sure that the tables chosen support error handling, either out-of-the box or via designing an appropriate business object with a lifecycle that supports an error status and the ability to resolve the error. Also note that the Sample Flat File Upload plug-in illustrates a header record / detail record scenario. In this case, the header record is linked to the child record via a CLOB element. This is not the recommended technique. In a real use case, the header record should be linked to the child record via a separate database column to allow for searching.

Configuring a New Process

The following points summarize the steps needed to implement a new file upload background process:

- Verify the details of the data in the upload file and map the data to fields in one or more appropriate tables in the system.
- Design the logic required for reading the record details and identifying each record to properly create the insert statements for storing the data. The sample plug-in scripts provided by the product illustrate using the various APIs available for use. Create a plug-in script where the algorithm entity is **Batch Control - File Upload**. Create an appropriate algorithm type and algorithm for this plug-in script.
- Create a batch control by duplicating the base template. Plug in the algorithm created above and configure the parameters as appropriate. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the File Upload algorithm.

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.

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.

The system provides a REST web service (using the inbound web service **F1-SubmitJob**) to create a batch job submission request from an external system. Note that no parameter defaulting occurs when submitting a record using a web service. All the desired parameter values must be provided.

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 one or more algorithms that check some conditions to assess whether or not the process is performing as expected. An algorithm 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 algorithms for a given background process, if desired. Each 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 are any level of service algorithms configured, the logic is executed and the results are displayed. The level of service is also part of the Health Check service.

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.**

Use the **Batch Control Query** portal to search for an existing batch control. Once a batch control is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Enter the **Application Service** that is used to provide security for submission requests for the batch control. The application service must have an Access Mode of **Execute**. Refer to Granting Access To Batch Submission for more information.

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 algorithms for the batch control. If one algorithm is plugged into the Batch Control, the level of service lookup value along with a message indicating the reason for the output value is shown. If multiple algorithms are plugged in, the text **See Results for Details** is displayed. There is an icon provided to expand the details returned by each algorithm. 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. Any value entered here is displayed to a user when submitting a job through online Batch Submission. Users may override the value before submitting the job. This includes resetting the value for an optional parameter. Regardless of how a batch job is submitted, if a value is not supplied for a required parameter, the default value define here, if found, is used. Optional parameters with no value supplied at submission time will not be defaulted to a value defined here. 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
Error Post-Processing	Optional	Algorithms of this type are called if the batch process fails to complete due to an error. Multiple algorithms are allowed and are executed in sequence order. Refer to Error Post-Processing Logic for more information.
File Upload	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to upload all records for a file. Note that only one algorithm is allowed.
Level of Service	Optional	Algorithms of this type are called to determine the Level of Service provided by a batch control. Multiple algorithms are allowed. Refer to Assessing Level of Service for more information.
Post-Processing	Optional	Algorithms of this type are called after all threads are complete. Multiple algorithms are allowed and are executed in sequence

System Event	Optional / Required	Description
		order. Refer to Post-Processing Logic for more information.
Process Record	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to process a specific record.
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.

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**.

Defining File Integration Types

This section describes the functionality supported for viewing and maintaining file integration types.

Refer to Extract Record Processing for an overview of the file integration type functionality provided with plug-in driven background processes.

To view or add a file integration type, open **Admin** > **Integration** > **File Integration Type.**

Once a file integration type record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **File Integration Records** zone lists the record definitions for this integration, in hierarchical order. Use the zone actions to add, maintain or view a file integration record. Press the **Record Layout** icon to view a pop-up display of the record's schema.

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. Note that batch jobs initiated by DBMS scheduler may also be put in canceled status. The system will look for records in this status and look for batch runs that may need to be canceled for this batch job. Once it is confirmed that any batch runs are stopped, the status changes to **Ended**.

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 by a user online.

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.

Granting Access To Batch Submission

Every batch control must reference an application service. When you link a batch control to an application service, you are granting all users in the group the ability to submit the associated batch job for execution.

> **FASTPATH:** Refer to The Big Picture Of Application Security for information about granting users access rights to an application service.

When the batch control is added to the batch job table, the system checks if both the user submitting the job and the user ID recorded on the batch submission record have access rights. Application security does not apply when viewing a batch control or an associated batch run.

> **NOTE:** Base batch controls will be associated with base owned application services by default. Implementations will need to ensure the appropriate user groups are linked to these application services.

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.

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. It may differ from the user who submitted the batch, as described below. 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.

Language 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 **Submission Method** indicates how the batch job was initiated. Valid methods include:

- **Online** - populated when a user manually creates or manually duplicates a batch job submission.
- **Generated** - populated by algorithms that submit a batch job and by 'initiator' batch jobs that submit other batch jobs.

- **Scheduled** - populated by the DBMS scheduler. Note that for other schedulers, it is not advised to use this value as there may be logic and validation related to records created with this value that may not apply to other schedulers.
- **Timed** - populated by the batch daemon that submits timed batch jobs.
- **Other** - populated when no other value is provided.

Additional values may be added by your implementation. Refer to the lookup **SUBMIT_METH_FLG** for a list of the submission methods supported by your system.

The **Submission User** indicates who initiated the batch job. This value is not populated for timed batch jobs or batch jobs submitted via a scheduler. It is only displayed if the field is populated.

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 **Start Date/Time** and **End Date/Time** of the batch run are shown.

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: Compression. The log files are compressed and include a *.gz extension. Different browsers treat this type of file differently. Some browsers may automatically decompress the file as part of the download and they are viewable in any text viewer by changing the extension to one the text viewer recognizes. However, some browsers download the compressed file and a user will need to unzip the file prior to viewing.

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:

- **Last Update Timestamp** contains the date and time the most recent batch run started or 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**.

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.

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.

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** > **System** > **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.

External Reference

Attachment records may be added from an external system in which case, the external system could have a reference ID to capture. The attachment allows for including an external reference ID when adding an attachment internally.

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. To do this, you may use the generic attachment zone provided by the product: **F1-ATTCHOWN**.

Note that your product may already have support for viewing and maintaining owned attachments on one or more if its base delivered portals.

Configuring Size Limit

The system allows for configuration of a size limit for uploading attachments. This is optional but the recommendation is to set this to a reasonable value so as to limit the ability to upload unreasonably big files. To configure this, navigate to Feature Configuration. Search for an existing feature configuration with the Feature Type **General System Configuration**. If one does not exist, create a feature configuration of this type. Select the **Maximum Attachment Size** option and define an appropriate value.

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.

Supporting File Name Exceptions

By default, the system prevents attachment file names from having special characters. If there is some reason that an implementation needs to define an attachment with special characters, there is a system property setting that may be configured to relax this validation. Contact your system administrator for support.

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.

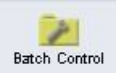
Maintenance Object Button

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. Note that only service programs that are Java based or Java (converted) are applicable here. There are some Services in the application generated for portals. There is no XML for those services.

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.

Groovy Java Docs Button

The **Groovy Java Docs** button switches to the Groovy 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.

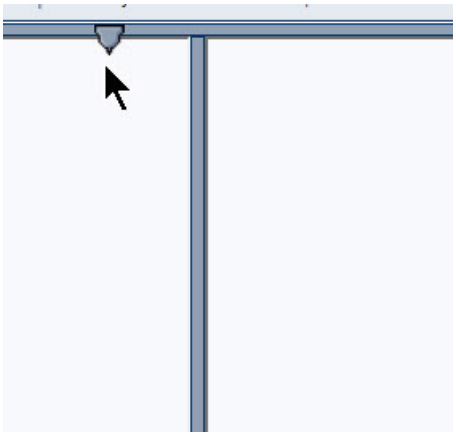
About Button

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.

Groovy Java Docs Viewer

The Groovy Java Docs viewer is an interactive tool that allows you to browse Java documentation files (Javadocs) for the Java classes that are accessible to Groovy code within scripts.

> **NOTE:** For system protection, only a subset of the base Java classes are available for use by Groovy code.

To open the Groovy Java Docs viewer from within the application viewer, click the Groovy Java Docs button. You can also access the viewer via the ‘View Groovy Javadocs’ link in the context sensitive Script Tips zone. Refer to the additional topics in the Java Docs Viewer section for details of how to navigate the viewer panels.

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
defaultLanguage	The default language used when the application viewer is started. Available values are those marked as language enabled on the language page.
defaultView	The default view then the application viewer is started. Available values include: - Data Dictionary
dataDictionary	Whether the Data Dictionary is available or not: - Y - N
sourceCode	This property is not being used. Simply enter 'N'.
baseHelpLocation	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
appViewerHelp	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 baseHelpLocation. 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.

Monitoring and Analytics

This chapter describes various tools provided in the product to support system monitoring including tools related to reporting, monitoring and analytics.

Reporting Tool Integration

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. Note that the product includes an out-of-the-box integration with BI Publisher. However it is possible to use the reporting objects to integrate with a different third party tool.

Integration with BI Publisher

Your DBMS, your product, and BI Publisher 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.

Multi-Language and Localization Support

The integration supports a multi-language implementation and supports different localization settings.:

- All labels, headings and messages are defined using field and message meta-data in the application, which support multiple languages.
- The appropriate font, size, and layout are based on the requested report and the user's language.
- Dates and numbers are formatted as per the user's display profile.
- Currency based numbers are formatted as per the currency definition from the product

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.

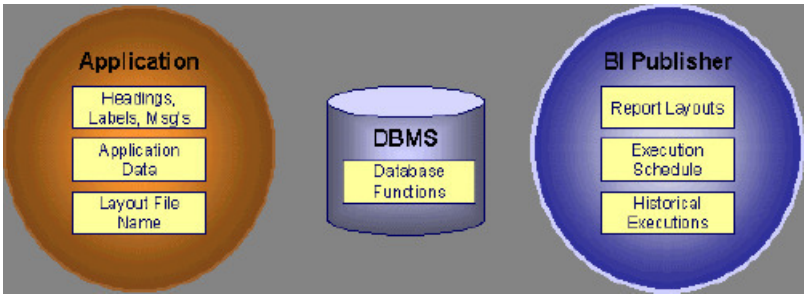


Figure 5: Application and BI Publisher

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 *Installation Guide* or *Optional Products Installation Guide* for information on this configuration.

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 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

If your reporting tools supports it, the Report History page may be configured to open tool’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.

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

Reporting Folder defines the shared folder where reports are stored.

Reporting Server From Browser 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>`.

> **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**.

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 CM_TODO is named CM_TODO_Base.

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 your product's *Installation Guide* or the *Optional Products Installation Guide*.

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. The following fields display:

Parameter Code is the identifier of the parameter. This must correspond to the parameter definition in the reporting tool.

Required indicates that a value for the parameter must be defined when submitting the report.

Sort 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 indicates the characteristic type used to define this parameter.

Default Value is option and if populated is displayed to the user when the report is chosen on the report submission page.

Description is a brief description of the parameter. This description is used when displaying the parameter on the report submission page.

Long Description is 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

Refer to the *Installation Guide* or *Optional Products Installation Guide* for 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.
- Defining a user for integration with your product.
- Publishing the sample reports in BI Publisher.

Contact your system administrator to verify that the above steps have occurred.

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. Refer to the documentation for this products for details about publishing a report. Refer to Publishing Reports in BI Publisher 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.

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/	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_GLTY
P_TAX_ACCTY_CHAR	Account Type Char Value for Tax Related GL Account	CI_GLTY	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.

Service Health Check

The system provides a plug-in spot on Installation, called Health Check. The algorithms plugged in here are used to check various system conditions and return details to help assess the health of the system.

What Information is Checked?

Each algorithm is responsible for checking one type of health check condition. The condition should be defined as a valid health component type (as defined in the **HEALTH_COMP_FLG** lookup). The algorithm may check conditions across many records (referred to as Health Components) and return a status and details for each health component. For example, if the algorithm is checking the level of service for batch programs in the system, the batch control is the health component in this case. The algorithm should gather the level of service results for each batch program and return the collection of information to the calling program.

The algorithms for the plug-in spot should return the following information for each health component.

- The **Health Component Detail** should be populated with information specific to the object that was checked. For example, if the algorithm is checking batch control level of service, this will have the Batch Control code. In addition, for enabling display for details of each health component type, the algorithm should populate the maintenance object code and primary key field and value details along with the navigation option.
- The **Health Component Status Flag** and **Health Component Status Description** should be populated with a valid status code and description appropriate for the condition being checked. This information could differ based on the health component type as each health component type may have its own unique status values.
- The **Health Component Status Reason** is meant to provide supporting information about the health component's status value. For example, if the component provides an error type of status, this could have the status reason.

- The **Health Component Response** is used to map the status value of the health component to a standard value defined in the lookup **HEALTH_RESPONSE_FLG**. The values are a subset of HTTP response codes. The supported values are **All Checks Successful** (200); **Non-Critical Function Degraded** (203), **No Content** (204) and **One or More Critical Functions Degraded** (500).
- The **Response Details** list is provided for health components that check several conditions as part of the health check. Each condition could return its own status. In this situation, the algorithm should use the response details list to record the individual responses and populate the overall Health Component Status Flag, Description and Reason with summary information. The suggestion is to do the following:
  - Set the status flag and response based on the details of the individual responses. Refer to the batch level of service information below for an example of this logic.
  - When the response list is populated, the health check user interface will show the overall Status Reason text along with an icon to expand the details. The suggestion is to populate this field with text like "See Results for Details".

The system provides an algorithm that checks the **Batch Level of Service** health component type. This health component type finds all the batch controls that are configured with at least one level of service algorithm and invokes the algorithms 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 based on whether the batch control has one algorithm or multiple. If there is one algorithm, the Level of Service lookup value returned by the algorithm is returned. If there are multiple, the system determines an overall status based on the detailed status values from each algorithm. If any algorithm returns Error, that value is returned. Otherwise, if any return a Warning, that value is returned. Otherwise Normal is returned.
- The **Status Reason** is populated based on whether the batch control has one algorithm or multiple. If there is one algorithm, the expanded text of the status reason returned by the algorithm is returned. If there are multiple, the text from message category / message number 11002 / 22001 is returned (**See Results for Details**)
- The **Response** is populated based on the value of the overall 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**.

How are the Algorithms Called?

These algorithms are called by a business service provided by the product **F1-HealthCheck**.

That service calculates an overall Health Response value based on all the details returned by all the algorithms. The values described above for the **HEALTH_RESPONSE_FLG** are the ones used.

Also note that the system provides Inbound Web Services for this business service for both SOAP and REST service calls allowing external systems to use a web service to retrieve this information.

Health Check Configuration

To enable your system to display health check information, navigate to the Installation Options - Algorithms collection and add an entry for the **Health Check** system event for each health check algorithm provided by the product that your product would like to include in your system health check assessment.

Viewing Health Check Results

The product provides a portal that allows a user to view the detailed results of the health check business 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.

The grid displays the detail of each individual component checked as part of the system health check. Refer to Service Health Check for more information about the details returned.

Introducing Health Check Conditions

If your implementation would like to check a condition for system health that is not already provided by your product, here are the steps required:

- Define a new value in the Health Component Type lookup (**HEALTH_COMP_FLG**).
- Provide a new algorithm type, plug-in script and algorithm for the Health Check system event that checks the condition that you wish to check. Refer to the Service Health Check topic for the information that each algorithm should be supplying for the condition. Note that if you follow the pattern of the system delivered Batch Control - Level of Service algorithm, your algorithm may be designed to check conditions across many records of the same type and gather all the results.
- Plug this new algorithm into the Installation Options - Algorithms collection using the **Health Check** system event.

Measuring Performance

Many implementations need the ability to track and view the performance of key system processes against a defined target level. The framework provides objects to allow an edge product or an implementation to calculate and display performance measures against a desired target for one or more use cases. The topics in this section provide information about what is supplied in the framework and guidelines for implementing a specific use case for batch processes. Your product may supply out of the box support for additional use cases. Refer to your product documentation for more information.

Understanding Performance Targets

The following are examples of use cases that would be well suited for tracking as performance targets:

- Track and view the duration of key batch processes, either individually or as a group, and how they relate to a defined target.
- Monitor used and free space on a database against critical levels.
- Check the performance of individual user interface zones against a defined performance expectation
- Compare the number of web service requests made to an application against a threshold where performance may be of concern.

Framework provides functionality to define and categorize performance targets and link them to objects such as business services, zones and portals. This supports the calculation and display of the metrics against desired results.

In addition, Framework supplies out of the box support for batch process performance targets. Individual edge applications may supply more specific functionality for other use cases, if applicable.

Ideally, users should have the ability to view these performance targets on a dashboard that groups related measures. Framework provides the necessary components to achieve this for batch process performance targets.

The following sections highlight functionality supported for performance targets in the framework. Refer to the edge application product documentation for more details of other supported use cases.

Performance Target Objects Overview

The setup of a performance target involves a unique combination of configuration data and processing logic that calculates and displays a specific measure.

The framework performance target functionality is supported by a combination of inter-related objects, as shown below. Some of these objects will be generic for use in all performance targets while some are specific to a functional area such as batch processes.

- **Maintenance Objects** for capturing performance target types and performance target instances.
- **Extendable Lookups** to define performance target categories and performance target metrics.
- **Business Services** to calculate known metrics for a group of performance targets, such as for batch processes.
- **UI Maps** to interpret the performance calculation results and display them in charts that show the comparison to the target.
- A **Zone** that serves as a template for system duplication to create specific zones related to each performance target. The zone invokes a business service to perform calculations and display performance measures in the related UI map.
- **Business Objects** to capture configuration data for a specific performance target instance and its related objects.
- Functionality to create **Zone instances** for specific performance targets based on the associated template zone.

Each edge application will deliver the following to compliment the objects delivered by the framework:

- **Portals** to group related performance target zones.
- Specific entries for the performance target category **Extendable Lookup** .

The following sections describe the combined use of these objects for performance targets in more detail.

Calculating and Displaying Performance Targets

Performance targets are intended to be displayed in a portal using an explicit object zone. The zone parameters define both the business service used to calculate the performance metrics and the UI map that displays the results. While an individual performance target needs to reference a zone with a unique configuration that calculates a particular metric, those zones will be based on a template zone which defines the core parameters.

Framework provides a base Batch Performance Target Metric Template zone (**F1-PERFBA**) for batch process performance targets. Refer to this zone and its parameters for more information about the batch performance zone configuration and the related business service and UI map.

Performance Target Metrics and Metric Types

The framework supports two types of metrics for performance monitoring:

- **Value based metrics** are used to record results against a specific numeric target.
- **Time based metrics** are used to track the results against a specific date and time target.

The list of valid metrics for a given performance target category and its associated performance target types is maintained using an extendable lookup. Framework uses the base business object Batch Performance Target Metric (**F1-BatchPerfTargetMetric**) to define batch process metric values. Refer to this lookup for the supported batch process metrics.

> **NOTE:** While users are not prevented from adding new values to the lookup, the list is not intended to be extendable as new values will not be recognized by the business service that performs the base batch performance calculation logic.

Your edge product or implementation may supply other extendable lookup business objects for additional performance measurement use cases, if applicable.

Performance Target Categories and Types

There are key configuration details required by all performance targets. These are defined on two related objects.

Performance Target Categories

Target categories define the template zone and security setting for a group of performance measures. The list of valid categories is maintained using an extendable lookup.

The framework product supplies the business object Performance Target Category (**F1-PerformanceTargetCategory**) for this functionality. Refer to the business object description and configuration for more information.

Performance Target Types

Target types define the related performance target business object and the display portal for a group of performance measures. In addition, a target type references a target category which defines the associated zone details.

The framework product supplies the business object Performance Target Type (**F1-PerformanceTargetType**) for this functionality. Refer to the business object description and configuration for more information.

The framework does not deliver any standard type or category values for batch processing performance targets. Refer to your specific edge application products to verify if any standard values are delivered for the batch processes within your applications. Edge applications may also supply standard categories and types for additional performance target use cases.

Performance Targets Define Specific Metrics

Although the types of measures and the business services and UI maps that govern how they are calculated and displayed are defined using separate objects, a **performance target** record defines the additional configuration needed to measure a specific metric and compare the result to a desired value.

There are key configuration details required by all performance target instances. These include a reference to the metric being measured, the desired target value or time, the desired result for the target and the unique zone by which this performance target will be monitored.

> **NOTE:** The performance target maintenance object has a direct foreign key link to the extendable lookup business object and value that define the performance metric. This is an unusual pattern as extendable lookup values are normally recorded only in an object's schema. The pattern has been adopted to allow the description of the metric to be displayed in the performance target maintenance portal.

A given **performance target type** may require additional details for its calculations. For example, the framework batch process performance target defines additional details to restrict the measurement to specific batch processes that have executed within a given time frame. These details are configured on the performance target business object. Refer to the embedded help text on the business objects supplied by the framework and your edge applications for more information.

The **performance target type** also defines the business object to use when creating the resulting performance target record.

Objects Linked to a Performance Target

Some performance measures such as batch process metrics, derive the data for the calculation from objects within the system. The performance target can be linked to one or more related objects to define the specific data sources included in that target metric. For example, when creating a batch process performance target, the framework supports linking specific batch codes to the performance target to indicate the group of batch processes that are included in the measure.

The performance target business object may be configured to allow only relevant objects types to be linked to a performance target record. Refer to the base Batch Performance Target business object (**F1-BatchPerformanceTarget**) for an example.

Creating Performance Target Zones

Once a performance target and its objects have been defined, a unique zone needs to be created to display and monitor the specific target.

The framework functionality for batch performance targets implements the zone creation via the related business object lifecycle. When a performance target is added, status enter plug-ins are responsible for generating the new zone using the template zone and prefix configured on the target category (or the override values) and adding the zone to the portal configured on the target type. When a performance target is inactivated, an enter plug in is responsible for removing the zone from the portal.

NOTE: While the template zone associated with the performance target category may be overridden, the zone generation algorithm makes certain assumptions about the zone type and parameters. In particular, the logic expects to configure a zone parameter that references the performance target code as input to the business service responsible for calculating the metrics.

Refer to the base Batch Performance Target business object (**F1-BatchPerformanceTarget**) for details of the lifecycle and the enter plug-ins responsible for performance target zone creation.

Setting Up Performance Target Configuration

The following topics highlight the general configuration steps required to use performance target functionality. Your particular product or implementation may supply additional functionality to support specific use cases for performance targets. Refer to your product's documentation and the library of business objects supplied for performance target in your system for more information.

Performance Target Category Lookup

Refer to About Performance Targets for an overview of performance target functionality.

Each Performance Target Type is associated with a unique Performance Target Category. The categories are defined as an extendable lookup.

Navigate to the Extendable Lookup portal. Search for and select the **Performance Target Category** business object. Define values that are appropriate to the categories your implementation is assigning for the performance target types in use. In some cases, an edge application may already have delivered the appropriate performance target category for your use

Refer to the embedded help for more information about configuring this object.

Defining Performance Target Types

Refer to About Performance Targets for an overview of performance target functionality.

To maintain the performance target types applicable to your product or implementation, open **Admin** > **Reports** > **Performance Target Type.**

This is a standard All-in-One portal.

The information captured on the performance target type depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Maintaining Performance Targets

This section describes the functionality supported for viewing and maintaining performance targets.

Refer to About Performance Targets for an overview of performance target functionality.

Navigate using **Main** > **Reporting** > **Performance Target**. This is an All-in-One portal and includes the standard List and display zones for a performance target.

For information about the data defined on the performance target, refer to embedded help.

Capturing Statistics

The product provides objects to allow for an edge product or an implementation to periodically calculate and capture statistics for one or more use cases. The topics in this section provide information about what is supplied in the framework and guidelines for implementing specific use cases.

Understanding Statistics

There are two objects that work together to capture the statistics related to a given business use case:

- The statistics control record which defines configuration related to capturing statistics. It also defines a category that is used to group similar types of statistics together.
- The statistics snapshot is the record where the actual calculated statistics are captured.

Statistics Control

The statistics control record defines whether or not statistics are automatically calculated and how frequently. In addition, you may control the retention policy of snapshot record. You may indicate that only the most recent snapshot is kept or all snapshot records are kept or that they are retained for a defined number of days.

The system provides a business object for statistics control (**F1-Statistics**) that is expected to be used for most use cases. Its lifecycle includes the support for periodic capturing of statistics snapshot records as well as on demand capture.

Statistics Snapshot

The statistics snapshot object is the object that is responsible for calculating and capturing the statistics details. A separate business object must be defined for each use case. The schema of the business object defines the details that are captured. In addition, it must include an algorithm as an enter plug-in on the **Complete** state that is responsible for capturing statistics.

The system provides a 'root' business object for statistics snapshot (**F1-SnapshotRoot**) which includes the lifecycle that statistics snapshot business objects should follow. No explicit statistics snapshot use case is provided by the Framework. Your specific product may supply some out of the box support for certain use cases, in which case specific statistics snapshot business objects are provided. Refer to your product specific documentation for more information.

Viewing / Reporting Statistics

The product provides the mechanism for defining statistics control and statistics snapshot records and viewing the data in basic format. If your product has provided support for specific use cases, there may also be additional portals used to display the statistics in a meaningful format. For example, the statistics snapshot may be capturing data that can be presented in graphical format. Additionally, if multiple historical statistics snapshot records are retained, a zone may be defined to graph changes over time.

Configuring Your System for Statistics

If your product provides statistics use cases that you are planning to implement, all that you need to do is configure the appropriate statistics control record and define the appropriate configuration for your business requirements. Refer to Defining and Monitoring Statistics for more information.

If your implementation has identified an additional use case where you would like to capture statistics, the following points highlight the steps needed to configure the system to support the use case.

- Determine whether an additional Statistics Category value is needed. This is captured on the statistics category record. Navigate to the Lookup page. Search for and select the **STAT_CATEGORY_FLG** field. Review the values and determine if additional values are needed.
- Define a new statistics snapshot BO. This should be a child BO of the base delivered BO (**F1-SnapshotRoot**). Its schema should define elements for the specific information that is captured by the statistics calculations. The schema should be designed in conjunction with an appropriate Enter Status algorithm for the **Complete** that calculates the statistics as appropriate for the business requirement.
- Once the snapshot business object is designed and implemented, configure the appropriate statistics control record.

Defining and Monitoring Statistics

Refer to About Statistics for an overview of the statistics functionality.

To define and maintain statistics control records and view a list of the current snapshot records, open **Admin** > **Reporting** > **Statistics Control.**

This is an All-in-One portal and includes the standard List and display zones for a statistics control.

For information about the data defined on the statistics control, refer to embedded help.

Statistics Snapshot

If there are any Statistics Snapshot records for the statistics control, the **Statistics Snapshot List** zone displays a list of the most recent records. A user may drill into any of the records to view more detail. You are brought to a maintenance portal where you may view details about the calculated statistics. The information captured will be unique to the particular use case. Refer to the embedded help for more information.

Creating Cube Views

Many implementations require the ability to extract and analyze complex sets of data in a way that is simple to visualize and supports business or organizational decision making. A common model for performing this type of analysis is a multi-dimensional array, often referred to as a data cube. Cube views allow data to be represented in a way that provides the user with multiple perspectives of the results.

Although called a "cube", a cube view can have two, three or a higher number of dimensions. Each dimension represents some attribute in the database and the cells in the data cube represent a measure or value. For example, a cell could contain a count of the number of times that attribute combination occurs in the database, or the minimum, maximum, sum or average value of data related to that attribute. Queries are performed on the cube to retrieve information.

The framework provides objects to support the definition of data cubes and a flexible means of viewing them. The topics in this section provide information about what is supplied in the framework to assist in implementing a cube view. Your product may also supply out of the box support for relevant use cases. Refer to your product documentation for more information.

Understanding Cube Viewer

The framework provides a cube viewer portal which is intended to be used for most implementations of cube views. It supports a number of common actions and data representations.

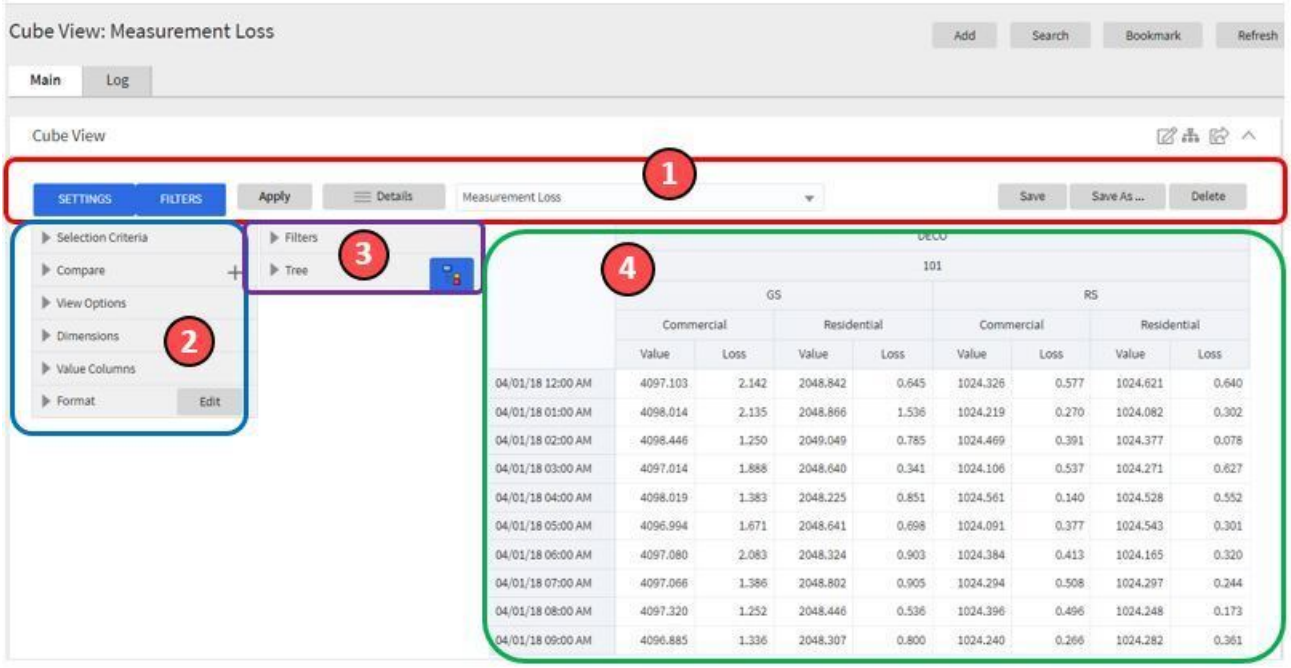
> **NOTE:** The choice of appropriate menu for the Cube Viewer portal will differ from product to product. Refer to your product specific documentation for more information on how to access the viewer.

The following sections describe the basic concepts and features of the cube viewer.

Cube Viewer Components

The base Cube Viewer interface is comprised of the following main areas:

1. The Toolbar
2. The Settings Area
3. The Filters Area
4. The Views Area



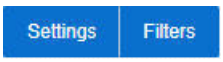
The next topics provide more information about each component.

The Toolbar

This following features are available on the Cube Viewer toolbar.

Settings and Filters Buttons

Click the **Settings** and **Filters** toggle buttons to collapse and expand the respective Settings or Filters areas.

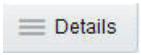


Apply Button

Click the **Apply** button to apply any changes to the settings and display the new results in the views.

Details Button

Click the **Details** button to open a dialog box in which additional attributes of the current cube view can be viewed and updated. Refer to Cube View Details for more information about the attributes displayed.

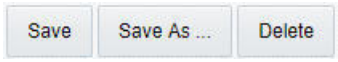


Cube View Dropdown

Click the **Cube View Dropdown** to open a list of cube views, of the same type as the current view, to which you have access. Selecting a cube view from the list will open that view. Use the search box to search for a view within the list using the cube view description.

Save, Save As and Delete Buttons

Click the **Save** button to save the current settings for the cube view. Click the **Save As** button to create a copy of the current cube view. Click the **Delete** button to delete the current cube view.



Cube View Details

Clicking the **Details** button on the cube viewer **Toolbar** allows you to maintain the following information for the cube view:

- The short **Description** for display in the viewer header and a more **Detailed Description** for capturing additional information about the cube view.
- An **Access Type** of shared, private or public.
- The **Access Group** that defines the group of users with access to the cube view. This field is only visible if the access type is shared.
- A specific **User** who has sole access to the cube view. This field is only visible if the access type is private.
- The **Row Functions** text box in which the functions to be applied to the values across the rows in the data grid are defined. Clicking anywhere in the box will display a list of supported functions to choose from.
- The **Column Functions** text box in which the functions to be applied to the values in the columns in the data grid are defined. Clicking anywhere in the box will display a list of supported functions to choose from.
- An **Inactivate** button. Click this button to deactivate this cube view.

The **Fixed Parameters** for the cube view are displayed but cannot be updated here. Refer to Configuring Cube Types for more information on cube view parameters.

The Settings Area

The Settings Area contains a number of sections whose inputs control various aspects of the current cube view.

Selection Criteria

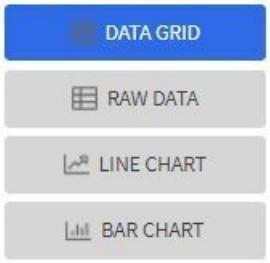
Open the **Selection Criteria** section to enter filter values for the user parameters defined on this cube type. User parameters that have been configured as 'Save with Cube View' on the cube type will be saved when the cube view is saved.

Compare

Open the **Compare** section to enter one or more sets of criteria that are used to retrieve data sets for comparison to the data retrieved by the main Selection Criteria. Use the Add and Delete buttons to add or remove a set of criteria. The comparison values will be displayed side by side with the main cube values in the data views. Note that the pseudo dimension **Dataset** must be included as a dimension column in order to display comparison views.

View Options

Open the **View Options** section to access buttons that toggle on and off the display of the supported data view types. The buttons can be used to change the order in which the views are displayed, using drag and drop. Any changes made to the default button settings will be saved when the cube view is saved. Refer to the Views Area topic for more information on the available data views.



Dimensions

Open the **Dimensions** section to view the list of dimensions defined on this cube type, as well as the pseudo dimension **Dataset** and the date/time filter, if applicable. Use drag and drop to move a dimension into the list of **Columns** or **Rows** displayed in the data views. Moving a dimension into the **Unassigned** list will remove it from the data views.

Format

Click the Edit button on the **Format** section to open a dialog box in which conditional formatting, such as background or text color, can be defined for the values in the data grid. Refer to Defining Conditional Formatting for more information about the formatting controls.

NOTE: Changes made to the settings will only be reflected in the views when the **Apply** button is pressed, with the exception of the **View Options**, where changes are reflected immediately upon toggling or rearranging the buttons.

Defining Conditional Formatting

Clicking the **Edit** button in the **Format** section of the **Settings** area allows you to define formatting rules for the display of values in the cube view data grid. The rules are based on mathematical expressions or formulas which use the cube view values as variables.

The formulas must be expressed using the field names of the values for the cube, as used in the cube view query zone. A list displaying the **Column Name** and associated **Label** for each value configured in the cube view is provided, to assist in defining the formulas.

The following attributes need to be defined for each rule:

- The **Sequence** defines the order in which the formatting rules are evaluated.
- The **Active** check box indicates whether the rule should be applied or not..
- The **Formula** defines the mathematical expression that is evaluated to determine whether the formatting should be applied. The expression variables should reference the **Column Name** of the value.
- **Format** defines the formats to be applied to cells whose value meets the condition expressed in the formula. The formats include the **Background** and **Text** colors and whether the text should be in **Bold**.
- The **Stop If True** check box indicates whether to stop evaluating the remaining formulas in sequence if this formula evaluates to true.

The Filters Area

The Filters Area provides the ability to select and apply dimension filters to the current view.

The **Tree** section displays a hierarchical view of the dimension values in the current cube view. Clicking a value will cause it to be displayed in the **Filters** section and immediately applies that filter to the data in the currently selected views. To remove a dimension from the filters, click on the value in the **Filters** section.

The **Hierarchical Filtering** icon, , is used to toggle between hierarchical and non-hierarchical selection mode. In hierarchical mode, clicking a dimension value at any level in the tree will automatically populate the filters with the related higher level dimension values. In non-hierarchical mode, only the selected dimension value is added as a filter.

Use the expand and collapse icons, , to quickly expand the tree, one level at a time, below the currently selected dimension value or collapse the entire level.

The Views Area

This section describes the types of views supported by the Cube Viewer.

Data Grid

The **Data Grid** displays the cube view data set in the form of a pivot table. The fields that appear as rows and columns in the table are defined in the **Dimensions** section of the Settings area. The default aggregation function used for the values is sum. Additional functions can be defined in the details of the cube view. Refer to the description of the **Details** pop-up window in theToolbar area for more information.

Raw Data

The **Raw Data** view displays the cube view data set as a flat table.

Line Chart

The **Line Chart** displays cube view data in a chart with a time sequence as the X-axis and value intervals as the Y-axis. Each line corresponds to a specific combination of the selected dimensions. This view is only visible if the cube view includes at least one date/time filter in the selection criteria. If the criteria include more than one date/time field, the first date/time filter in order is chosen for the time sequence.

Bar Chart

The **Bar Chart** also displays cube view data in a chart with a time sequence and value intervals. Each bar corresponds to a specific combination of the selected dimensions. This view is only visible if the cube view includes at least one date/time filter in the selection criteria. If the criteria include more than one date/time field, the first date/time filter in order is chosen for the time sequence.

Cube Configuration Components

Cube views are supported using a number of standard configuration components.

The system provides a business object for cube views (**F1-CubeView**) which is used to capture access controls and the settings, such as user parameter values, for a cube view instance. The business object also defines the portal navigation and the base display UI Map (**F1-CubeViewDisp**) and accompanying display map service script (**F1_CubeViewD**). The display map and service script together drive the functionality supported in the cube viewer.

> **NOTE:** The system also supplies a UI map fragment (**F1-CubeViewProc**) which is used to display a cube view in a process flow panel. Refer to Process Flows for more information.

The system provides a portal for cube views (**F1CUVWM**) that can be configured on the appropriate menu item or other navigation path for your implementation. The portal is linked to the base cube view zone (**F1-CUBEVIEW**).

In addition, the system supplies a simple query by cube view information (**F1-CUBVIWQ1**) that is the single entry on a base multi-query zone (**F1-CUBVIWQ**). The multi-query zone is not linked to the base portal as the expectation is that implementations will need to create searches tailored to their specific use cases for cube views. The base multi-query provides a quick means of configuring a cube view search while in development mode.

Configuring Cube Types

Cube Types are used to capture the details that define the sourcing of the data for a cube view and parameters that control certain features of the cube viewer. These details include:

- The business service whose associated zone defines the query used to select and filter the data for the cube view.
- The user parameters that correspond to the filters for this cube type's business service and zone.
- The user parameter settings that determine whether a given filter value should be saved with the cube view and whether the value should be populated from global or portal context, if applicable.
- The fixed parameter values to be applied as hidden filters for cube views of this type.
- The results from the business service and zone to be used as dimensions in the cube view and those that are the cube view values.
- The dimension settings that determine the order in which the dimensions are displayed in the data grid, which dimensions are to be used as filters in the cube view and which dimensions are to be included in the cube view tree.
- The values settings that determine the default sequence in which the values are displayed in the data grid.
- The FK Reference used to navigate from the data grid to the underlying source data.

If your product provides business services that support cube views you plan to use, all you need to do is configure the appropriate cube type records. Refer to Maintaining Cube Types for more information.

If your implementation has identified additional cube views you need for data analysis, you will need to create a zone and business service that use specific techniques for defining the SQL to derive the source data for the view and for mapping the results. Refer to Cube Type Advanced Topics for more information

Maintaining Cube Types

Refer to Creating Cube Views for an overview of the cube viewer functionality.

To define and maintain cube type records, open **Admin** > **Analytics Configuration** > **Cube Type.**

This is an All-in-One portal and includes the standard list and display zones for a cube type.

For information about the settings defined on the cube type, refer to Configuring Cube Types and the embedded help.

Cube Type Advanced Topics

The following topics describe the technical models that need to be followed when creating a new cube type.

Designing Your Cube Type SQL

In order to support the base cube viewer, the SQL used in your cube type data explorer zone needs to be constructed with two main components. The first component creates a view of the data and its dimensions in a form that supports a tree-like structure and allows for the dynamic use of filters. The second component queries the filtered data set to provide the values to be displayed for each dimension.

The first component should contain the following queries:

- A sub-query constructed using a 'with' clause that builds a tree-like view of the cube's dimensions and optionally other columns used in the main SQL. This portion of the SQL must be enclosed within '/*>tree*/ ' and '/*<tree*/ ' hints and use the table alias **C_TREE**. The combination of the dimension columns should form a unique 'key' in the **C_TREE** data. Both the 'with' clause and the 'tree' portion within it may contain other supporting sub-queries.
- A sub-query which is the placeholder for filtered queries on the 'tree' data. It must use the table alias **C_FILTER** and take the form 'C_FILTER as (select * from C_TREE where 1=1 and ROWNUM <= n)'. The value of 'n' should be set to limit the rows returned to a number that is manageable within the cube viewer.

The second component is the main query that retrieves the measures for the cube view dimensions. This query references **C_FILTER** as the subset of the **C_TREE** and joins with other tables if required to retrieve the numeric values. The first date or date/time result column of the main query (as specified in the zone configuration) designates a time-series column, which forms the time-series 'key' with the dimensions columns.

Defining Your Cube Type Business Service

The business service that identifies the zone for a cube type is also used to provide additional configuration options.

The business service schema should define meaningful element names for the parameters, dimensions and values. It should also define labels for those elements to be used by both the cube type and cube viewer user interfaces. The labels can be derived by mapping the element to a meta-data fields or by using the '**label=**' syntax to define a text string directly in the schema definition. The '**required=true**' syntax can be used to require a value to be entered for a user parameter on the cube viewer or for a fixed parameter on the cube type.

The user parameter elements are expected be defined within a group in the schema called 'input' and the fixed parameters within a group called 'hidden'.

Product Use Metrics

The Product Use Metrics portal includes a zone (or zones) that are used to perform a business review of your organization's implementation of the product to confirm licensing / subscription compliance. The information visible on this portal will depend on the product or products installed by your organization.

To view this portal, open**Main** > **Dashboards** > **Product Use Metrics.**

For more information about what each zone on the portal is reporting, refer to the embedded help.

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.

Understanding Inbound Web Services

The system supports communicating with the system via RESTful services or via SOAP services. In both cases, the system uses an object called inbound web service (IWS) to store the configuration. A web service class is used to distinguish whether the IWS is used for REST or SOAP. There is also a distinct business object for each web service class value because there is different configuration for each type of IWS.

For both types of inbound web service, the system 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.

The following topics provide more information about the different configuration of IWS based on whether it is SOAP or REST.

Inbound REST Web Services

The following topics provide more information about the product's support of REST services using IWS.

HTTP Method and Parameters

When defining operations for inbound REST web services, the product supports the HTTP methods of Get, Patch, Post, Put, and Delete. Note that the product's support of these various HTTP methods are a means of communicating the purpose of the web service to the outside world. However, the actual behavior of the REST web service is driven by the behavior of the underlying schema based object (business object, business service or service script). For example, you may configure the HTTP Method of "Put" for an operation that references a service script that just retrieves a record. The product is not able to

detect this type of discrepancy. Configuration users should carefully consider the correct method to use based on the logic of that service.

For operations that reference a business object, the transaction type must be provided. The REST syntax doesn't support defining the transaction type at runtime. There are some basic validation checks in this case related to the transaction type and the HTTP method. For example, a Get method only makes sense with the Read 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.

For the HTTP Methods of Get, Patch, Put, and Delete you may additionally define parameters. For each parameter, you define an external reference to the element, which is how this parameter is exposed to the external callers and is defined in the API specification. Each of these parameters is mapped to the XPath of the schema element from the underlying business object, business service or service script. For each parameter, you indicate if it is a **Path** parameter or a **Query** parameter.

- Path parameters are parameters that are part of the endpoint and are required. Each path parameter must be included in the operation's URI component surrounded by curly brackets.
- Query parameters are optional. They are not part of the endpoint but rather are included in the endpoint URL after a question mark, followed by name value pairs.

Refer to the URL section below for examples of path and query parameters in the sample URLs.

URL Composition

When building the endpoint URL for a REST service, there are three main parts that make up the full URL.

- The first part is the one that is environment specific. This will differ for an on premise implementation as compared to a cloud implementation. Both will have the host and the port and then additional components that identify the environment.
- The second part is a hard coded path designated by the product, namely "/rest/apis".

  These two parts of the URL are defined in the substitution variable **F1_REST_BASE_URL**. This is defined when initializing your environment. Refer to the Server Administration Guide for more information.

The remaining part of the URL is built dynamically based on configuration for each IWS and its operations. The components are "/ownerURIComponent/resourceCategoryURIComponent/iwsURIComponent/operationURIComponent"

- The owner URI component is taken from the owner flag of the inbound REST web service. A special extendable lookup (Owner Configuration for REST Services) defines this component for each owner flag.
- Each Inbound REST Web Service must reference a resource category. This category is used to associate related web services to a common category of resources. For multiple IWS records linked to the same resource category, external catalogues can use this information to group together related web services. The resource category is an extendable lookup and the URI component is an attribute of this record.
- Each REST IWS record defines a URI component that serves as an external identifier of this IWS record. The value must be unique within a given owner flag.
- Each operation must define the HTTP Method and optionally a URI component. When defining path parameters for the Get, Put or Patch HTTP Method the path parameters must be included in the URI component using curly braces.

In all cases, the URI component must begin with a slash ('/').

The following are some examples of the dynamic portion of the URL for a REST service. The last example illustrates the use of a query parameter.

IWS Name URI Component	Owner URI Component	Resource Category URI Component	Operation HTTP Method URI Component	Dynamic URL Component	Run Time Example
c1rateCalculation /rateCalculation	C1 /customer	C1-RATES /rates	Post /	../customer/rates/ rateCalculation/	../customer/rates/ rateCalculation/
w1WorkActivity /workActivity	W1 /asset	W1-WORK /work	Get /{activityId}	../asset/work/ workActivity/ {activityId}	../asset/work/ workActivity/5798165498
w1WorkActivity /workActivity	W1 /asset	W1-WORK /work	Patch /scheduleWindow/ {externalSystem}/ {activityId}/ {windowStartDateTime}	../asset/work/ workActivity/ scheduleWindow/ {externalSystem}/ {activityId}/ {windowStartDateTime}	../asset/work/ workActivity/ scheduleWindow/ MY-COMPANY/5798165498/20190101
CM-AccountActivityHistory / accountActivityHistory	CM /cm	CM-ACCT-INFO /accountInformation	Get /{accountId}	../cm/ accountInformation/ accountActivityHistory/ {accountId}	../cm/ accountInformation/ accountActivityHistory/123456789? activityId=5468976

Inbound SOAP Web Services

For inbound SOAP web services, by default the system 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 XSLs to transform into the relevant schema formats. In addition, if desired, the Request and Response Schemas that document the expected message may be provided.

NOTE: The Request and Response Schema fields are not supported in a cloud implementation.

NOTE: The system supports defining the XSLs as a managed content record. For backward compatibility, the system supports defining the XSL as a file in the file system. This is a system wide setting, defined using a feature configuration option. The feature type is **External Messages** and the option type is **XSL Location**. Set the value to **F1FL** to support XSL in the file system. To support the XSL in managed content, no option is needed as this is the default. You may explicitly define the value of **F1MC** if desired.

In addition, note that 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 SOAP 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 Java Enterprise Edition (Java EE) Web Application Server. This allows for multiple policies to be attached as supported by the Java EE 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 Java EE 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. The base product provides annotation types that support the standard WS-Policy (**F1POLICY**) and OWSM Security Policy (**F1-OWSM**). No base annotation is supplied by the product for either annotation type.

  If your implementation wishes for the policy of a particular IWS to be attached at the Java EE Web Application Server, define a special annotation for the **F1POLICY** annotation type and configure the **uri** parameter value to **<none>**.

NOTE: Refer to WebLogic documentation for more information on supported security policies.

NOTE: In order to use the OWSM Policy, additional system configuration is necessary. Contact your system administrator to confirm if your implementation supports OWSM.

Inbound SOAP Web Service Deployment

A Inbound SOAP Web Service must be deployed to the Java EE Web Application Server in order for it to be available to the Web Service Clients to access the system. Refer to Deploying SOAP 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 Java EE 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 Java EE 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 SOAP 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.

NOTE: There is no support for XAI inbound services via REST, only via inbound SOAP web services.

Configuring Inbound Web Service Options

This topic describes the configuration needed for using inbound web services.

Configuring SOAP Inbound Web Service Options

This topics in this section describes the configuration needed for using inbound SOAP 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 some base annotation types. Refer to the metadata for more information. 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 Java EE Web Application Server are not sufficient for your implementation's needs.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service annotation type code. Refer to System Data Naming Convention for more information.

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.

The system supports the ability for an IWS record to refer to multiple policies. In this situation, the annotation type for the policy should include a reference to a parent annotation type so that the system can properly build the array of annotations.

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.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service annotation code. Refer to System Data Naming Convention for more information.

Maintaining XAI Inbound Services

XAI Inbound Service is a legacy mechanism for exposing system services to external systems as web services. The current supported mechanism for defining web services is Inbound Web Services, which should be used for any new web service.

Some framework based products and existing implementations may have existing XAI inbound services that may need to be viewed. For these services, the product supports deploying them as SOAP inbound web services for execution. Refer to Deploying XAI Inbound Service via IWS.

The following sections describe basic information about the maintenance pages. Note that some of the information is related to legacy functionality that is no longer supported.

XAI Inbound Service - Main

To view an inbound service, open **Admin** > **XAI** > **XAI Inbound Service**.

Description of Page

XAI In Service Name 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 **XAI Service ID** serves as the primary key.

Owner indicates if this XAI inbound service is owned by the base package or by your implementation (**Customer Modification**). This information is display-only.

The **Adapter** defines the interface with the target application server.

If adapter for this service invokes a system service, then the appropriate **Service Name** is visible.

If adapter is the **Business Adapter** then **Schema Type** and **Schema Name** reference the object to invoke.

Web Service Category is visible if the XAI inbound service is linked to one or more web service categories.

Use the **Description** and **Long Description** to describe the service.

Check the **Active** switch if this service is enabled and available for execution.

Post Error is not applicable.

Trace is not applicable.

Debug is not applicable.

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 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.

XAI Inbound Service - Staging

The configuration on the staging tab is no longer supported.

XAI Inbound Service - Parameters

The configuration on the parameters tab is no longer supported.

Configuring REST Inbound Web Service Options

This topics in this section describes the configuration needed for using inbound REST web services.

Owner Configuration Lookup

Refer to Inbound REST Web Service for an overview of REST IWS functionality.

When generating the URL for a REST web service, part of the URL is taken from the owner of the Inbound REST Web Service record. The component to add to the URL is defined in this lookup using the URI Component field. The values are provided by the product.

To view the values, navigate to the Extendable Lookup portal. Search for and select the **Owner Configuration for REST Services** business object.

Resource Category Lookup

Refer to Inbound REST Web Service for an overview of REST IWS functionality.

Each Inbound REST Web Service is associated with a Resource Category. The categories are defined as an extendable lookup. The REST URL is built using a component from the IWS record's resource category.

Navigate to the Extendable Lookup portal. Search for and select the **Resource Category** business object. Define values as needed. Typically the category is related to a logical grouping of entities, or maintenance objects, referred to as a 'resource' in REST nomenclature. In some cases, an edge application may have already delivered appropriate resource categories for your use.

Populate the appropriate URI component to include when building the URL for IWS records linked to this resource category.

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 support for REST and SOAP services and provides several inbound web services out of the box.

For inbound SOAP web services, 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.

CAUTION: Important! When adding new records, carefully consider the naming convention of the inbound web service code. Refer to System Data Naming Convention for more information.

The **Inbound Web Service** zone displays the configuration information for the record and its operations.

Refer to Inbound SOAP Web Services for specific information about Operations for SOAP records.

Refer to Inbound REST Web Services for specific information about Operations for REST records.

Note that in addition to standard actions available on this portal, there is also a special **Add to Category** button in the page action area. Click this button to link an inbound web service to one or more web service categories.

Deploying Inbound SOAP Web Services

Once an Inbound SOAP 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 SOAP Web Services to the Java EE 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 Java EE 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 Java EE Web Application Server. This can also be done manually for clustered deployments, if desired.

There are two methods available for deploying inbound SOAP web services:

- Deployment at the command line using the **iwsdeploy[.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 SOAP Web Services

The Deploy Inbound SOAP 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 SOAP 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 Deployment**.

- 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 SOAP 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 Deployment**. 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 Undeployment**. 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.

NOTE: For cloud installations, the system supports defining the WSDL URI via a substitution value defined for the token URI: @F1_BASE_IWS_URI@. The system will use this value per the functionality described in URI Substitution. When no value is defined, the system uses a default URL, which is the appropriate URL for on-premise installations.

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 Deployment**. 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 Undeployment**. 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 Deployment" 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.

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 do the following:

- 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.

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: OUAF Outbound Message that may be used by an implementation to define messages to process and how to process them. This transport processes outbound messages in order of the priority defined on the outbound message type.

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 a persisted message is desired, the recommendation is to use the Outbound Message Dispatcher. This business service creates the message using standard BO processing, relying on the outbound message logic to route the message and store the record. The message is routed after the BO pre-processing algorithm and after the record is persisted but before the BO post-processing and audit plug-ins are executed.
  - If the message should not be persisted, then the recommendation is to use the Outbound Message Mediator. As mentioned, the Outbound Message Dispatcher creates the outbound message record, relying on the outbound message logic to route the message. If it should not be persisted, it is subsequently deleted. In contrast the Outbound Message Mediator executes the BO pre-processing algorithms and then routes the message explicitly without creating a message record. It is more efficient for scenarios that don't require persistence. Note that the Outbound Message Mediator also supports persistence, but it does so by creating the records without using BO processing. This is not recommended. The Dispatcher is the better option if persistence is desired.

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 Business Object and Type

The product supplies many outbound message Business Objects along with Outbound Message Types for out of the box functionality.

In addition, implementations may need to define configuration for custom outbound messages. 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.

Referencing an XSL

The system supports defining the XSLs as a managed content record. For backward compatibility, the system supports defining the XSL as a file in the file system. This is a system wide setting, defined using a feature configuration option. The feature type is **External Messages** and the option type is **XSL Location**. Set the value to **F1FL** to support XSL in the file

system. To support the XSL in managed content, no option is needed as this is the default. You may explicitly define the value of **F1MC** if desired.

Capturing the Outbound Message ID in the Message

If your integration would like to use the system generated Outbound Message ID as a unique identifier with the external system, the following configuration is needed:

- Define an element within the XML Source that should be populated with the system generated outbound message.
- Configure a BO Option on the outbound message's business object using the Option Type **Outbound Message ID XPath** and set the Option Value to the XPath of the element defined to capture the ID.

NOTE: This functionality is only applicable if the outbound message is persisted.

Support for Dynamic URLs

The product supports the ability to build a dynamic URLs. These are cases where the URL requires information determined at runtime. This is supported with a combination of BO schema definition, URL configuration and appropriate code when creating the outbound message. The following points highlight the steps needed to support this functionality.

- When defining the Message Sender's URL, use the syntax **${pathParms}** in the location of the URL where runtime information must be inserted. For example: **http://my.server.com:1000/rest/services/${pathParms}**
- Include the data area **F1-OM-DynamicConfig** (Outbound message dynamic configuration) in the schema of the business object for the outbound message. This data area includes elements for **pathParms** and **queryParms**.
- In the code that creates the outbound message, populate the **pathParms** and if applicable **queryParms** elements with the appropriate information. The system will build the URL by plugging in the **pathParms** element value followed by a question mark, followed by the **queryParms** element value into the **${pathParms}** location in the URL.

The following is an example of run time values.

```
   <dynamicConfiguration>
       <pathParms>job/1234</pathParms>
       <queryParms>firstName=John&lastName=Doe</queryParms>
    <dynamicConfigureation>
```

Add SOAP Header Parameters at Runtime

The product supports the ability to add SOAP header parameters to an external message at runtime. The following points highlight the steps needed to support this functionality.

- Include the data area **F1-OM-DynamicConfig** (Outbound message dynamic configuration) in the schema of the business object for the outbound message. This data area includes the element **soapHeaders** .
- In the code that creates the outbound message, populate the **soapHeaders** element with the self contained XML to add to the SOAP header section of the outgoing SOAP request.

Real-Time Message Configuration

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 **RTEMAILSNDR** 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.
- For HTTP senders including JSON senders, the system provides support for sending messages secured by OAuth authentication using Oracle Web Services Manager (OWSM). The system provides a pre-configured set of policies for OAuth (**F1-OAUTH**) using a special extendable lookup. Note that the values of this policy set defines a specific CSF key repository that the implementation should use for capturing its CSF keys. In addition, there is a substitution value defined for the token URI: @F1_OAUTH2_URI@. Configure the appropriate URI for this implementation as described in URI Substitution. By default the system does not support additional policy sets to be defined. If your implementation requires a different policy set, contact support.

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 with SOAP format automatically applied, use the **SOAPSNDR** - HTTP SOAP sender class. For routing via HTTP using JSON format, use the **RTJSONSNDR** - JSON sender class. For routing via JMS, use the **RTJMSQSNDR** - JMS queue sender class or

RTJMSTSNDR - JMS topic 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.

> **NOTE:** Refer to Support for Dynamic URLs for configuration needed to support dynamic URLs when sending an outbound message. There is specific configuration expected when defining the URL on the Sender in order to support dynamic URLs.

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 use for validation.

You may turn on or off this validation checking using the Message option **Schema Validation Flag**.

This feature is not supported for cloud implementations.

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. A variable may be used in place of all or part of the URL. The variable must be predefined in the substitution variable property file. The value here should enclose the variable name with @. Refer to Referencing URIs for more information.

Indicate the **Initial Context Factory**, which is a Java class name used by the JNDI server provider to create JNDI context objects.

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

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.

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.

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 as follows:

Message Class	Description
RTEMAILSNDR	Email sender.
RTHTTPSNDR	HTTP sender.
RTJMSQSNDR	JMS queue sender.
RTJMSTSNDR	JMS topic sender.
RTJSONSNDR	HTTP JSON sender.
SOAPSNDR	HTTP SOAP sender.

The following sender classes are related to MPL processing and remain in the product for upgrade purposes:

Message Class	Description
DWNSTGSNDR	Download Staging sender.
EMAILSENDER	Email sender.
FLATFILESNDR	Flat file sender.
HTTPSNDR	HTTP sender.
JMSSENDER	JMS Queue sender.
OUTMSGSNDR	Outbound Message sender.
STGSENDER	Staging Upload sender.
TPCSNDR	JMS Topic sender.
UPLDERRHNDLR	Upload Error Handler.

Indicate whether or not this sender is currently **Active**.

For JMS related senders, indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**. (Note that for all other types of senders, the Character Encoding context type is used to configure 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.

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.
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**, **RTJSONSNDR** or **SOAPSNDR**.

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. Refer to Support for Dynamic URLs for configuration needed to support dynamic URLs when sending an outbound message.	
HTTP Method	The HTTP method used to send the message. **NOTE:** The SOAP sender message class **SOAPSNDR** only supports the **POST** method.	**POST** or **GET**
HTTP Transport Method	Specifies the type of the message. You can either send the message or send and wait for a response.	**Send** or **sendReceive**
HTTP Form Data	Used when the message is in the format of an HTML Form (`Content-Type: application/x-www-form-urlencoded`). This context specifies the form parameters (data) that should be passed in the HTTP message. Since a form may have multiple	

Context Type	Description	Values
	parameters, you should add a context record for each form parameter. The value of a form parameter takes the format of *x*=*y* where *x* is the form parameter name and *y* is its value. If *y* contains the string @XMLMSG@ (case sensitive) then this string is replaced by the content of the service response XML message. The @XMLMSG@ string can be used in the HTTP Form Data or in the HTTP URL, but not in both. 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 @XMLMSG@ is not specified in one of the context records. If a context record of this type is not defined for a sender, then the XML is sent with `Content-Type: text/plain`. When using POST it is put in the HTTP message body. 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. The MPL server builds formData by concatenating the individual parts. You may use substitution variables when entering values for Form Data.	
HTTP Login User	The HTTP server may require authentication. Add a context record of this type to specify the login user to use.	
HTTP Login Password	The HTTP server may require authentication. Add a context record of this type to specify the login password to use.	
HTTP Header	Sometimes the HTTP server on the other side may require the addition of HTTP headers to the message. 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*	**UTF-8** or **UTF-16**

Context Type	Description	Values
	is the value of this context and when sending the message it will encode the data in that encoding.	
Response Time Out	The amount of time the system should wait for the remote system to send a response.	
Sender Security Type	Indicate the desired security type to apply. (Note that additional security types are applicable only to the SOAP Sender. See below for more information.)	**BASIC** (HTTP Basic), **TEXT** (Username Token plain text), **DIGEST** - Username Token Digest, **OWSM** - OAuth Security via OWSM,
OWSM Policy Set	Applicable only if the Sender Security Type is **OWSM**. Defines the policy set to apply.	Enter a valid value for the extendable lookup Set of Policies (**F1-SetOfPolicies**). The product provides the value **F1-OAUTH** that may be used here.

Real-time HTTP Sender

The following context type is only applicable to senders with the **RTHTTPSNDR** message class.

Context Type	Description
Content Type	Populate a value here to override the **Content-Type** attribute in the HTTP header, which defaults to **text/xml**.

SOAP Sender

A SOAP sender is an HTTP sender that automatically adds support for the SOAP format. For this type of sender (message class of **SOAPSNDR**), besides the context values listed above, the following context entries may also be supplied.

Context Type	Description	Values
Message Namespace URI	Used to indicate a specific namespace to be included in messages for this sender. Note that this value is used only when the External Message link to this sender is configured with a **Namespace Option** of **Configured on Sender**.	
SOAP Insert Timestamp	Indicate whether a timestamp should be added. Default value is 'N'.	**Y** or **N**
SOAP Expiration Delay (in seconds)	Indicate an expiration delay to add to the timestamp. The default value is 60.	
Sender Security Type	Additional security types are supported for SOAP senders for specific third party integrations.	**CIM** - Uses an appropriate namespace and adds security information in the SOAP header specific to Landis and Gyr, **MS_V30** - Uses an appropriate namespace and adds security information in the SOAP header specific to MultiSpeak v3.0, **MS_V41R** - Uses an appropriate namespace and adds security information in the SOAP header specific to MultiSpeak v4.1.**ITRON** - Uses appropriate

Context Type	Description	Values
		SOAP header information security information in the SOAP header specific to Itron.

NOTE: Refer to Add SOAP Header Parameters at Runtime for information about dynamically including SOAP Header parameters when sending a message.

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** or **RTJMSTSNDR**, respectively.

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

Refer to Outbound Messages for an overview of this functionality.

Use this page to define basic information about an outbound message type. Open this page using **Admin** > **Integration** > **Outbound Message Type.** You are brought to a query portal with options for searching for outbound message types.

NOTE: This page is not available if the **External Message** module is turned off.

Once an outbound message type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Outbound Message Type** zone provides basic information about an outbound message type. Besides the code and description, the outbound message type captures the outbound message business object and the relative priority of the outbound message. The priority is only applicable if the outbound message is processed using OSB. In addition, if the outbound message type is linked to any web service categories, they are displayed.

CAUTION: Important! When adding new records, carefully consider the naming convention of the outbound message type code. Refer to System Data Naming Convention for more information.

Note that in addition to standard actions available on this portal, there is also a special **Add to Category** button in the page action area. Click this button to link an outbound message type to one or more web service categories.

The **External System Links** zone is visible if the outbound message type is linked to any external systems.

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 and 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**.

Namespace Option is used when your message should include a namespace in the resulting XML. The valid option are **Standard Namespace** and **Configured on Sender**. If the value is Standard Namespace, the system will generate a namespace for the resulting WSDL with the following value: **http://ouaf.oracle.com/outbound/AAA_BBB**, where AAA is the external system code and BBB is the outbound message type code. If the value is Configured on Sender, then the value of the namespace is taken from the sender context with context type **Message Namespace URI**.

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**. Refer to Referencing an XSL for information about configuring where the XSL is defined.

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**. This feature is not available for cloud implementations.

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 only be displayed when the processing method is **Real-time**. Refer to Real Time Messages for more information on how it is used. Refer to Referencing an XSL for information about configuring where the XSL is defined.

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.

To define options for your environment, open **Admin** > **Integration** > **Message Option**.

Description of Page

The following options are supported.

Option	Description	Option Name
Default Email Sender	This is the default Message Sender used for sending e-mails when no explicit Message Sender is specified.	**defaultEmailSender**
Default User	The default user is used by the system to access your product when no other user is explicitly specified.	**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**
Messages Language	The default language to use for the messages.	**language**
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	**xaiOuboundSchemaLoc**

Option	Description	Option Name
	message schemas. For example: http:/ localhost/cisxai/schemas.	
Schema Validation Flag	Enter **Y** to turn on schema validation for outbound messages. Enter **N** to turn this off.	**xaiSchemaValidationCheck**
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**

Note that the following options are no longer applicable.

Option
Automatically Attempt Resend to Unavailable Sender (Y/N)
Default Response Character Encoding
JDBC Connection Pool Max size
Maximum Errors for a Sender
Messages JDBC Connection
MPL Administrator Port
MPL HTTP Server Authentication Method
MPL HTTP Server Password
MPL HTTP Server User
MPL Log File
MPL Trace File
MPL Trace Type
Privileged Users
Records MPL Receiver Will Process At a Time
Schema Directory
Send SOAP Fault as HTTP 500
Sender Retry Seconds
System Error JDBC Connection
System Error Max Retry
System Error Retry Interval
Thread Pool Initial Size
Thread Pool Max Size
Thread Pool Non Activity Time
WSDL Service Address Location
XAI Authentication Password
XAI Authentication User
XAI Trace File
XAI Trace Type
XSD Compliance
XSL Directory

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 or**Real-time**. (The value of **XAI** is no longer supported). 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 no longer applicable..

The **Creation Date** indicates the date that this record was created.

The status is no longer applicable.

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. 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: 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.

- The business service supports sending attachments. The may either reference attachments from the file system (file name and content ID) or reference attachment records captured in the system's attachment table. Note that attachments in the file system are not supported in a cloud implementation.

  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.

- 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.

Web Service Category

The product provides the ability to categorize web services that are defined in the system. The term web services refers to Outbound Message Types, Inbound Web Services, and for those implementations where they are applicable, XAI Inbound Services.

A given web service may be linked to more than one category. The product supplies web service categories and most base delivered web services are configured with appropriate categories. Implementations may define new categories and may link custom web services to base delivered categories or to custom categories. In addition, implementations may link additional categories to base delivered web services. However, implementations may not remove base delivered web services from base delivered categories.

If your implementation uses the Integration Cloud Service Catalog, the integration provides the categories for each web service that is sent to the catalog.

The web service category portal supports adding and removing web services from the category. In addition, the Outbound Message Type portal and Inbound Web Service portal each provide a page action button to **Add to Category**.

Defining Web Service Categories

Refer to Web Service Category for an overview of this functionality.

Use this page to define basic information about a web service category. Open this page using **Admin** > **Integration** > **Web Service Category.** You are brought to a query portal with options for searching for web service categories.

> **NOTE:** This page is not available if the **External Message** module is turned off.

Once a web service category has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Web Service Category** zone provides basic information about a web service category.

> **CAUTION:** Important! When adding new records, carefully consider the naming convention of the web service category code. Refer to System Data Naming Convention for more information.

Once a web service category exists, an **Add Web Services** zone is provided to find and select web services to link to the web service category.

When viewing a category that already has web services linked to it, they are visible in the zone **Included Web Services** . This zone allows a user to remove a customer owned web service from the list.

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 embedded help for information about the supported syntax.

The JMS Message Details zone displays a message broadcast from the list zone.

Oracle Integration Cloud Catalog

Oracle Integration Cloud (OIC) is an offering that serves as integration infrastructure for Oracle cloud solutions. The product provides an adapter for OIC to streamline integration between your edge application and OIC.

The product provides a mechanism (referred to as an adapter) for OIC to retrieve the REST catalog and another to retrieve the SOAP catalog from a given product. The expectation is that a given implementation would use either REST or SOAP and not both, but for those implementations that do support both classes of web services, OIC can retrieve both catalogs.

For the SOAP catalog, the adapter retrieves the name, the source system, the WSDL location and namespace for each web service.

For the REST catalog, the adapter retrieves the name, the source system and the OpenAPI specification for each web service.

It is possible that not every web service supported by an edge product is managed by OIC. In order to only include the appropriate web services in the adapter, configuration is needed to identify which web services to include.

The web service catalog is used to identify the records that should be retrieved by the adapter. Separate catalogs exist for REST and for SOAP

- For SOAP inbound messages, the system supports both the use of inbound web services and XAI inbound services that are deployed via IWS. Each IWS or XAI inbound service that should be included in the catalog must be flagged in the SOAP catalog. Note that only deployed services are returned to the catalogue.
- For REST inbound messages, each IWS that should be included in the catalog must be flagged in the REST catalog.
- For outbound message, the system requires the creation of an External System that includes each outbound message type that the external system receives. For outbound messages that are integrated through OIC, the external system itself will represent OIC. Rather than identifying each outbound message type to include in the catalog, only the external system needs to be flagged. The adapter will return web service information for all the outbound message types configured for the external system. If an implementation supports both REST and SOAP, separate external systems should be created to separately group the REST and SOAP outbound messages.

NOTE: Master and Subordinate Catalogs. The master and subordinate catalog functionality is no longer applicable and will be removed in a future release.

FASTPATH: Refer to Maintaining the Web Service Catalog for more information.

Web Service Catalog Configuration

The topics in this section describe the configuration needed in your edge application to integrate with OIC.

Web Service Catalog Master Configuration

The **Service Catalog Configuration** (master configuration) record defines several system wide settings related to integrating with the service catalog.

It includes configuration for the WSDL base URL for the SOAP catalog. Note that this element supports the functionality described in Referencing URIs. Note that for the REST catalog, the system will use the URL defined in the substitution variable **F1_OPEN_API_BASE_URL**

The subordinate server section is no longer applicable and will be removed in a future release.

For more information about specific fields in the master configuration, refer to the embedded help.

Maintaining the Web Service Catalog

Refer to Oracle Integration Cloud Catalog for an overview of web service catalog functionality.

To add or remove services that are reported to the OIC catalog, navigate to the portal using **Admin** > **Integration** > **Web Service Catalog.**

The Catalog by Class zone provides the list of classes (REST and SOAP). Broadcast the web service class that you want to work with.

The Catalog zone provides a list of services that are currently in the catalog based on the web service class chosen. Users may use this zone to remove services from the catalog.

The Candidate Services zone provides a list of external systems and inbound web services that are not currently linked to the catalog. Users may use this zone to add services to the catalog. The inbound web services will be limited to those with the same web service class as the one broadcast. All external systems are shown regardless of the web service class because there is no web service class configuration on external systems.. The expectation is that implementers will create the external system to include either only REST or only SOAP services and will know which external system to include in which catalog. For the SOAP web service class, if XAI inbound services are applicable, they are also visible.

NOTE: For inbound SOAP web services and XAI inbound services, they may be selected for the SOAP catalog at any time. However, only deployed services are returned to the catalog by the OIC adapter.

XAI Documentation Note

The XAI functionality is legacy functionality and not recommended for new implementations. The help topics are no longer provided.

Integrations

This chapter provides high level information about product integrations supported for all products that use Oracle Utilities Application Framework.

Creating Application Links in External Applications

When integrating Oracle Utilities applications with other external applications, you can create links from external applications to specific portals and pages within the Oracle Utilities applications (also known as "deep linking"). These links typically leverage Business Process Assistant (BPA) scripts to help navigate the user to the appropriate target portal or page and to make provided parameter values available for use on the portal or page.

NOTE: This technique may also be used within the product to launch a new browser tab from within the application when it's important to maintain the current screen. For example, launching the GIS Viewer in Oracle Utilities Work and Asset Management by creating a new browser tab.

To do this, parameters are added to the application URL by appending a question mark (?) to the end and using the syntax "key=value". If you require more than one parameter, use an ampersand (&) to separate each key=value pair. There are several predefined parameter key values recognized by the application. In addition, specific parameters understood by the script may be provided. The following list highlights the predefined parameters.

- **location=**<navigation option> - If the exact page to launch is known, the navigation option value of that page can be part of the URL using 'location='. For example, if an approval process has sent an email to a user to approve an adjustment, the link configured in the email may include the adjustment's navigation option.
- **script=**<BPA script name> - In almost every use case, a script is needed to help navigate to the correct page and display the correct information. Even if a location is known and provided in the URL, if there is context data provided as a parameter, it is the responsibility of a script to move that parameter data from the URL into the page data model, so that the page can access the information and display the correct record. In cases where the target page is dynamic based on information provided by parameters, the script may be used to determine what page to launch.
- **initNav=false** - By default, if a script is provided but the location attribute is not provided, the system navigates to the user's home page prior to executing the script. If the script itself includes a step to navigate to a target page as one of its initial steps, the navigation to the home page is unnecessary and may degrade performance. This optional attribute may be included in the URL to bypass the home page.

NOTE: The system still requires a page to be launched for technical reasons. A blank portal with no zones is used for this purpose. Users may see this portal (called **Launching Application**) briefly before the navigation initiated by the script. In addition, this is the portal that the user will remain on if there are any errors in the script or if the script does not navigate anywhere.

- **minDashboard=true** - This option minimizes the dashboard area as part of launching the application. (The user will be able to maximize the dashboard themselves if needed). You would use this option to potentially improve the speed at which the application is launched. Whether this is useful will depend on the use case. If the reason for launching the application is to bring the user to a specific record to view details about that record, the dashboard may be providing valuable information about the record. In this case, it may be correct to display the dashboard by default (and not include this parameter in the URL). On the other hand if the use case is that a user is already in the application and clicks a link that opens a new browser tab to show the user specific detail about a chosen record, while keeping the original page open on the first tab, the dashboard may contain the same information already visible on the original tab. In this case, minimizing the dashboard may be beneficial.

For example, the following URL illustrates launching the script **CM-StopAutoPay** for a given account, assuming the application URL for launching the system is http://system-server:1234/cis.jsp:

- **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).

Determining the Application URL

When building the link from within application logic, the service script **F1-EnviURL** may be used the retrieve and build the application URL information.

Navigate to a Given Record's Maintenance Portal

The product provides a script **F1-GotoPrtl** (Navigate to portal for an MO and key values) that may be used for any use case that simply navigates to the maintenance page for a given record and displays that record. This script is only applicable to records that are governed by a business object (and define a navigation option as a BO option). It requires the parameters **mo=** with the maintenance object code of the record and the parameters **pkValue1=** through **pkValue5** for providing the key values. It looks up the appropriate portal navigation option from the record's BO, populates keys into the 'page data model' and navigates to the portal.

It is also recommended to include the **initNav=false** attribute when building a URL with this script. The example below 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&initNav=false**

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 JDNI 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–LDAP** batch control can be updated with system wide configuration.

- JDNI 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

Element	Attribute	Description
		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 one of 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 the 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 eight 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	linkedToLDAPEntity	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) - **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=" (&amp;(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=" (&amp;(objectClass=organizationalUnit)(ou=%searchParm%))">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="name" cdxName="Group" />
      <LDAPCDXAttrMapping ldapAttr="description" cdxName=" Description" default="Unknown" />
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntity="User" linkingSearchFilter="
      (&amp;(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.

In order to use this functionality, feature configuration options for the **External Messages** feature type must be configured.

- Set option type **Support SPML Deployment in IWS** to **true**.
- Set option type **Default SPML service security policy** to an appropriate value per your implementation rules.

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 striations 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.

At a high level, the integration with the Oracle DBMS Scheduler supports the following entities:

- **DBMS Program.** A program should be defined for each Batch Control that needs to be scheduled by the DBMS scheduler. A program would typically invoke a batch job, but it could be configured to set certain options instead.
- **DBMS Chain.** A Chain defines a series of steps with dependency rules between them. A step references a program, with the program performing the actual work for that step. A rule is attached to each step to identify its dependent steps and the condition for when that step should be executed. For example, in a chain consisting of STEP_A and STEP_B, where STEP_B can only start if STEP_A was successful, the rule for STEP_B to start would specify a condition of "STEP_A SUCCEEDED".
- **DBMS Schedule.** A predefined frequency for jobs that need to be run periodically, for example, nightly jobs.
- **DBMS Job.** Defines a plan to perform a specific program or a chain periodically on a specific schedule or ad-hoc.

The product provides a set of business services to maintain these entities as well as submit jobs, manage submissions and report on past submissions. Refer to business services that start with the prefix "F1-DBMS" for more information.

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.

Understanding 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 synch 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.

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.

Operational Analytics

The framework provides several building blocks and tools that the edge applications may use to implement analytic reports directly from the operational application using Oracle Utilities Analytics Visualization product (referred to in this section as the analytics visualization product). The following sections provide more information about this functionality.

Understanding Operational Analytics

The following sections describe the type of configuration supported in your product to integrate with the analytics visualization product. Refer to the Oracle Utilities Analytics Visualization documentation for more information.

Direct Data Access

The analytics visualization product's canvases report directly off of the operational system thus eliminating the need of an ETL process. Selected tables and views have been designated as dimensions and facts for the purpose of generating the Start Schemas used by these canvases.

Calendar and Time Dimensions

Analytic reports rely on calendar and time dimensions to support various hierarchical grouping by date and time.

The Calendar dimension provides a level-based definition of the standard calendar and fiscal calendar in a flattened representation that is commonly used in data warehouses. This is necessary to enabling customers to group by calendar week, months, quarter and fiscal calendar period, quarter and fiscal year etc. Note that fiscal information about a specific date is optional and sourced from your edge application specific accounting calendar

In the same way, the Time dimension provides a level-based definition of each minute in a day, supporting reports that group by hours, AM/PM etc.

NOTE: Calendar dimension records are generated by a batch process; one record for each date in a specified period of time. You need to run the **Generate Calendar Dimension** batch process on an ongoing basis to cover the standard and accounting calendar days needed to support your business. Refer to **F1-BICDD** batch control for more information.

NOTE: Time dimension records are also generated by a batch process; one record for each minute in a day. You need to run the **Generate Time Dimension** batch process once to generate all the records. Refer to **F1-BITMD** batch control for more information.

Bucket Configuration

The analytics visualization 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 supports mapping of characteristics to user defined fields associated with dimensions in the analytics visualization product. Each characteristic table that is associated with a dimension table is provided with pre-generated user defined fields identified by unique column sequence numbers in the analytics visualization product. The mapping of characteristics to dimension user defined fields is maintained directly on the dimension portal. You may also review these mappings using the Characteristic Mapping portal.

Analytics Configuration

Depending on which product or products you have installed, there may be some configuration needed for the analytics visualization product. Refer to Defining Analytics Options for more information.

Calendar and Time Dimensions

This portal is used to view generated calendar and time dimension records.

Refer to Understanding Operational Analytics for an overview of operational analytic functionality.

Navigate using **Admin** > **Analytics** > **Calendar and Time Dimensions**.

The following zones may appear as part of the portal's **Main** tab page:

- **Calendar Dimension.** This zone lists most recent calendar dimension records. You may filter the list to review records generated up to a specific date.
- **Calendar Date.** This zone provides information about a selected calendar date.
- **Time Dimension.** This zone lists all time dimension records.

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.

Defining Analytics Options

Refer to Understanding Operational Analytics for an overview of the operational analytics functionality.

Your specific product may require some system wide configuration to be defined when using operational analytics.

Navigate to the Extendable Lookup portal. Search for and select the **Analytics Options** business object.

Use the embedded help to review information about the options to define.

Defining Analytics Tables

This portal is used to search for Analytics Table definitions. An Analytics Table record exists for each table or view in the operational system that represent dimension and fact tables in the Operational Analytics product.

Refer to Understanding Operational Analytics for more information.

You can access the portal from the**Admin Menu** > **Analytics Configuration** > **Analytics Table.** You are brought to a query portal with options for searching for a specific record. Once a record has been selected you are brought to its maintenance portal to view and maintain the selected record.

Defining Analytics Dimensions

This portal is used to maintain metadata for Analytics Dimension tables. A dimension record exists for each table in the operational system that represents a dimension table in the Operational Analytics product.

Refer to Understanding Operational Analytics for more information.

You can access the portal from the**Admin Menu** > **Analytics Configuration** > **Analytics Table.** You are brought to a query portal with options for searching for a specific record. Once a dimension record has been selected you are brought to this portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Table.** This zone provides information about the selected dimension record.

- **Fields.** This zone lists all the fields that define the dimension table. Fields may be selected from the dimension's underlying operation table using the **Excluded Fields** zone or added as computed fields.
- **Dimensions.** This zone lists dimension tables that this dimension references via foreign key constraints.
- **Referenced By.** This zone lists dimension and fact tables that reference this dimension via foreign key constraints.
- **Characteristic Mapping.** This zone manages characteristic mapping extensions for the dimension.
- **Excluded Fields.** This zone lists fields from the dimension's operational table that are not included in the dimension table.

NOTE: The ability to introduce custom analytics dimensions is currently not supported.

Defining Analytics Facts

This portal is used to maintain metadata for Analytics Fact tables. A fact record exists for each table or view in the operational system that represents a dimension table in the Operational Analytics product.

Refer to Understanding Operational Analytics for more information.

You can access the portal from the**Admin Menu** > **Analytics Configuration** > **Analytics Table.** You are brought to a query portal with options for searching for a specific record. Once a fact record has been selected you are brought to this portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Table.** This zone provides information about the selected fact record.
- **Fields.** This zone lists all the fields that define the fact table. Fields may be selected from the fact's underlying operation table or view using the **Excluded Fields** zone or added as computed fields.
- **Dimensions.** This zone lists dimension tables that this fact references via foreign key constraints.
- **Excluded Fields.** This zone lists fields from the fact's operational table or view that are not included in the fact table.

NOTE: The ability to introduce custom analytics facts is currently not supported.

Defining Characteristic Mapping

This portal maps characteristic extensions to user defined fields provided for each characteristic table that is associated with an analytics dimension table.

These mappings may be directly maintained on the operational Analytics Dimension portal. Refer to Understanding Operational Analytics for more information.

If your system is integrated with the Oracle Utilities Analytics product refer to Understanding Analytics Integration for more information about ETL based characteristic mapping.

You can access the portal from the **Admin Menu** > **Analytics Configuration** > **Characteristic Mapping.** You are brought to a query portal with options for searching for a specific mapping record. Once a record has been selected you are brought to the this portal to review the information for the selected record.

The **Characteristic Mapping** zone on the portal's **Main** tab page provides information about the mapping record.

NOTE: Only **Adhoc** and **Pre-defined** characteristic types are supported.

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.

Understanding 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 my 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.

ETL Based 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. The mapping is used by the extract process to populate target user defined columns with corresponding characteristic data.

This type of mapping 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.

Refer to Defining Characteristic Mapping 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.

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.

Understanding 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 Mobile 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.

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 both a REST and SOAP inbound web service that may be used for an external system to initiate (or update) a business flag. The REST service is **F1-BusinessInsights**. The SOAP service is **F1-BusinessFlagSync**. In both cases, the web service invokes a "driver" 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 "driver" service script includes a "contextData" group, which in turn references the data area **F1-BusinessFlagContext**. This data area is to allow for an external system to provide additional information for a given business flag use case. For any integration that requires additional information to be passed in from the external system, a separate data area should be used to define the additional elements; and that data area should refer to **F1-BusinessFlagContext** as its extended data area.

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, Oracle Utilities Analytics Insights or other machine learning platforms will aggregate and analyze large amounts of data to identify potential insights that need investigation. These platforms can initiate a business flag with a **Suspected** confidence.

An application receiving this business flag may adjust the confidence to either **Confirmed** or **Rejected**.

- • Setting the confidence to **Confirmed** is the equivalent of a true positive.
  - Setting the confidence to **Rejected** is the equivalent of a false positive.

Note that the application that receives a business flag is responsible for acting on the value based on business rules. Any updates to the confidence can be communicated to the other products that received the same business flag, if desired.

Additionally, creating a business flag in the application with a confidence of **Missed** provides a way to denote a missed insight. This is the equivalent of a false negative.

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 embedded 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 embedded help text for more information.

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 embedded help for more information.

Market Transaction Management

This section describes various entities and aspects involved in the configuration and maintenance of market transactions. The framework product supplies basic functionality to support the objects related to market transaction management. However more specific functionality (business objects and algorithms) for specific use cases are supplied by the edge applications.

Understanding Market Transaction Management

Several objects are provided to support market transaction management.

- Market Message Type. This object is used to define the different types of market messages, both inbound and outbound and to define the appropriate market message Business Object to use when creating objects of this type.
- Inbound Market Message. This object captures the instance of a market message received from an external source.
- Outbound Market Message. This object captures the instance of a market message that will initiate a message to an external source.
- Market Process Type. This object is used to define configuration related to processes that may be initiated to support steps that should be execute to support an inbound or an outbound market message. It defines the market process Business Object to use when creating market processes of this type. It may also define one or more market process event business objects to initiate events that are generated to support steps within the market process.
- Market Process. This object captures the instance of a market process that may be created to carry out business logic related to a specific inbound or an outbound market message.
- Market Process Event. This object is used to orchestrate individual events that may occur to support steps in a given market process.
- Market Configuration. This object may be used to capture configuration needed to support market transactions. Each type of configuration should use a Business Object to define what information needs to be captured. Each type of configuration may define up to five key fields that may be used to uniquely identify an individual configuration instance.

Configuring Market Transaction Management

The following sections describe market transaction management configuration options.

Defining Market Configurations

Each market configuration is used to define one or more options for a given type of configuration (based on a delivered business object).

The product defines a generic set of portals, described here, that may be used to define any type of market configuration. However, it may be that your implementation includes more specific configuration portals for certain types of market configurations.

Open this page using **Admin** > **Market Transaction Management** > **Market Configuration.**

You are brought to the **Market Configuration Query** where you need to search for the market configuration (i.e., its business object).

Once you have found the appropriate market configuration, select the value and you are brought to a standard All-in-One portal that lists the existing values for the market configuration. The standard actions for an All-in-One portal are available here.

Defining Market Message Types

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Admin** > **Market Transaction Management** > **Market Message Type.**

This is a standard All-in-One portal.

The information captured on the market message type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Defining Market Process Types

This page is used to define market process types which control behavior of the resulting market processes and the valid market process events that could be triggered as a market process progresses through its lifecycle.

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Admin** > **Market Transaction Management** > **Market Process Type.**

This is a standard All-in-One portal.

The information captured on the market process type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Maintaining Market Transactions

The following sections describe various pages available to view and work on market transactions.

Maintaining Market Processes

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Main** > **Market Transaction Management** > **Market Process.**You are brought to a query portal with options for searching for market processes.

Once a market process record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Market Process** zone provides basic information about a market process. Refer to the embedded help for more information.

Related Objects

The related object tab includes several zones that display various objects that may be related to the market process.

Related Market Messages and Processes

This zone displays any entries in the related object collection that are outbound market messages, inbound market messages or market processes.

Related Processes

This zone displays market processes that are linked to this market process using the explicit related process table. The market processes linked here are marked either as a parent, a child or a sibling market process.

Other Related Objects

This zone displays market process events linked to this market process. In addition, the zone displays any miscellaneous objects that are linked to the market process in the related objects collection, besides outbound market messages, inbound market messages and market processes.

Maintaining Market Process Events

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

You can view this page when drilling into a market process event from the Market Process related objects tab.

Once a market process record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Market Process Event** zone provides basic information about a market process. Refer to the embedded help for more information.

Related Objects

The related object tab includes a zone that displays all entries in the market process event's related object collection.

Maintaining Market Messages

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Main** > **Market Transaction Management** > **Market Message.**You are brought to a query portal with options for searching for market messages. Both inbound market messages and outbound market messages may be queried here.

Once a market message record has been selected, you are brought to the appropriate maintenance portal (inbound market message or outbound market message) to view and maintain the selected record. The inbound market message portal and the outbound market message portal have similar portal tabs and zones.

The **Market Message** zone provides basic information about the market message. Refer to the embedded help for more information.

Related Objects

The related object tab includes several zones that display various objects that may be related to the market process.

Related Market Messages and Processes

This zone displays any entries in the related object collection that are outbound market messages, inbound market messages or market processes.

Other Related Objects

This zone displays any miscellaneous objects that are linked to the inbound / outbound market message in the related objects collection, besides outbound market messages, inbound market messages and market processes.

Configuration Migration Assistant (CMA)

This chapter describes the Configuration Migration Assistant (CMA), a facility to enable the export of data from one environment to another.

CAUTION: This chapter is intended for users responsible for creating environments for testing.

Understanding CMA

The topics in this section provide a general understanding of the Configuration Migration Assistant (CMA) tool.

CMA Overview

CMA Process Overview

The Configuration Migration Assistant (CMA) provides implementers with a flexible, extensible facility for migrating 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.

The following points highlight some common use cases for using this tool:

- Seeding a newly created environment with configuration data. For example, if you are creating test environments, you can import configuration data from a development environment. In this case, you will probably set up a "copy all configuration" type of migration. Another use case is if you want to troubleshoot an issue in production, you may want to copy all production configuration to a test environment.
- Updating an existing test environment with new configuration or updates to configuration from a development environment. In this case your set of objects will be more targeted and you are doing more of an incremental migration.
- Promoting tested configuration to your production environment.

Note that in all cases, when records exist in both the target and the source, the CMA tool will fully replace the object with the version from the source. (Note that CMA does provide a way to Adjust Imported Data)

Also note that the CMA process **does not handle deletion** of records in target environment. When importing records from a source environment to a target environment, 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 orchestrate 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.

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.

CMA Configuration

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.

- System wide settings are defined 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.
- A **Migration Request** is used to define the data to include in a given migration. There are several types of migration requests:

FASTPATH: For more information, refer to CMA Configuration.

CMA Execution

Once you have a migration request defined to include desired data to export, the next steps are to export from the current environment and import to the target.

- The **Export** process includes all the steps needed to select records to be exported from the source environment and create the export file. 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 Additional Migration Considerations 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. For more information, refer to Importing and Applying a Migration.

Types of Migrations

The Configuration Migration Assistant is used for two general types of migrations: wholesale and targeted. The following topics 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. With the ability to group migration requests, the expectation is that implementations follow these guidelines:

- Multiple migration requests using the Criteria-based or Entity List migration request classes are used to group information logically together to allow for more reuse.
- A Group migration request is used for the export. This allows for one data set to export and one data set on the import side, simplifying the process. Note that depending on the amount of data, this may be a large import set to process. An implementation may find it easier to create multiple migration requests that break down the process into several steps.

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 can build a Group migration request that includes the base migration request and your custom migration requests to have a consolidated export.

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.

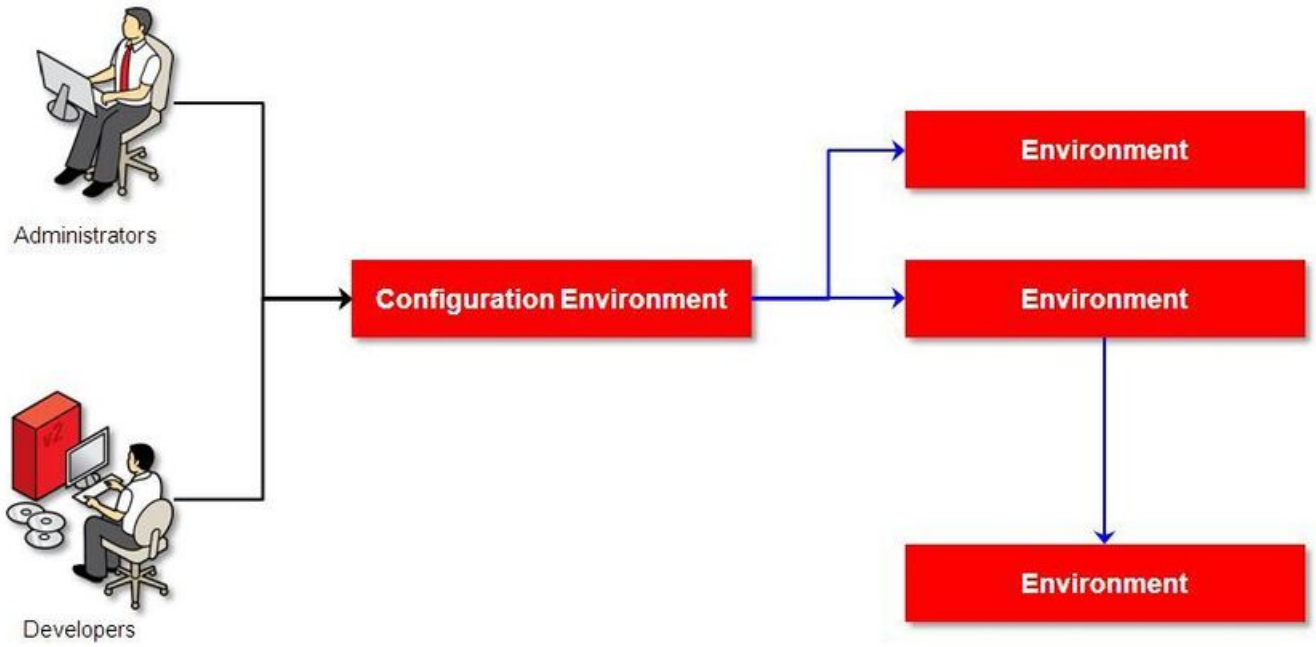
Targeted Migrations

A targeted migration refers to migrating a specific subset of data from one environment to another. Examples of targeted 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.

Migrating Configuration Data

A main use of CMA is to migrate configuration data, where the source environment is considered a kind of system of record and the target environment is getting updates based on changes to the source environment. Alternatively you may be seeding a new environment. The expectation is that the target record is updated to look like the source record.



Configuration data may include a mixture of records that have an owner flag and ones that do not. 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

For records with system generated keys, there is an issue if the target environment already has a record with the same key but doesn't represent the same record. The tool shouldn't be just updating that target record with the view of the source in case this will break foreign key relationships that exist in the target.

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 are based on the Attachment maintenance object. 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 data with system generated keys.

The product recommends that an implementation establishes a migration strategy such that 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

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 special logic to be used by any MO that may contain administrative data mixed in with master or transaction data. This special logic 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.

Importing Data that References a User

Although the tool supports copying users from one environment to another, the assumption is that it will be more common for implementations to define their valid users for each environment separately. In fact, the default CMA configuration for copying security configuration assumes that users are not being copied.

There may be times were a record is being copied from one environment to another where a user is referenced on the record but that user is not valid in the target environment. For example, if a batch control is configured as Timed with a user referenced, if this batch control is copied to another environment, what should happen if that user is not found? You may not want that user defined in the target environment. To handle this, the product provides support for a Default User that you can define in the migration assistant configuration. If this default user is configured then any time the CMA import process recognizes that a record references a User and the user does not exist in the target environment, it updates the record to instead refer to the default user.

NOTE: Please note that if you are trying to copy configuration that includes valid users, you should ensure that all the users are copied to the target environment first. Otherwise, the configuration will be updated to refer to the default user, which may not be your intention.

NOTE: If you copy records that reference a User within an XML field or CLOB, the CMA input tool is not able to detect that. A pre-compare algorithm is recommended if this use case exists in your implementation.

Additional Considerations

The following sections describe some additional considerations for CMA.

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 administration 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 doesn't 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

The **Migration Assistant Configuration**master configuration record defines system wide configuration used in both the source environment and the target environment.

As described in Importing Data that References a User, if you have records being imported that may reference a user does not exist (and shouldn't exist) in the target environment, define a Default User here.

In both the source environment and the target environment, the system needs to know the location of the export directory and the import directory along with the expected file suffix. For these fields, implementations may also rely on the system defaults. If no Migration Assistant Configuration record is found, the system assumes that there is an entry defined in the system's substitution variable list for **F1_CMA_FILES**. Further it defaults the values as follows:

- Export directory is the value for this variable plus **"\export"**.
- Import directory is the value of this variable plus **"\import"**.
- File suffix is set to **cma**

Refer to Referencing URIs for more information about the substitution variable list.

For additional information about specific fields in the master configuration, refer to the embedded 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 migration plan is always related to a given maintenance object, define using its primary instruction. Multiple migration plans may exist for a given maintenance object if there are different requirements for migrating records in that MO under different circumstances. For each migration plan, you decide if additional data should be included as part of an object using

subordinate instructions. For example, when copying a single algorithm type, should the system only copy that algorithm type record, or should it also copy all its algorithms and its related script (if applicable)?

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 an XML column to be referenced.

  NOTE: The #parent expressions may access elements that are stored in an XML column 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.
Import	Optional	Algorithms of this type are no longer supported.

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 XML-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. Defining instructions ensures that the related objects are included in the migration, if appropriate. If you know that the related objects are included because of other criteria, 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 purpose.

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 used to define the data to be included in a migration. 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.

There are three types or classes of migration request. The system provides a base business object for each along with a migration request class, which matches the business object. The subsequent sections provide more information about each class of migration request.

Note that all migration requests support defining a Category, which allows implementers to categorize the migration request.

In addition, all classes of migration request include the following zones:

- **Migration Request** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Referencing Migration Requests** This zone is only visible if the displayed migration request is included in a Group migration request. It lists each group migration request that includes it.

Other zones may appear for specific classes of migration requests. See the following sections for more information.

Criteria-based

This type of migration request defines a set of migration plans to be logically included together as part of a migration. For each migration plan, selection criteria is defined to indicate which records for each MO should be included. Selection may be defined using SQL, an algorithm or explicit primary key values.

- For selection using SQL Statement, refer to the embedded help for examples.
- For selection using Algorithm, the algorithms that are eligible to be plugged in here are those valid for the **Migration Request - Select** system event.
- For selection using Specific Keys, the primary key (1 through 5) must be explicitly specified. Multiple rows are allowed.

Entity List

This type of migration request allows the user to choose explicit MO / prime keys. The MOs that are eligible are those that are configured with a **Default Migration Plan** option. Although the user is managing MO / PKs, the migration instructions are still defined with a migration plan. The system maps the migration instructions in a similar way to a **Criteria-based** migration requests that use a Specific Key selection type. Note however that it will create a separate migration instruction for each MO / PK combination. It does not try to group all PKs for the same MO / migration under one migration instruction.

For this type of migration request, a user adds a migration request record with its description and other main information. Then, a special zone **Add Entities** is provided to find and select records based on a selected maintenance object and add to the migration request. The user is prompted to provide a reference and comments, if desired. If the category selected is one that requires a reference, then this will be validated.

When maintaining a migration request with existing entities, they are visible in the zone **Migration Request Entities** . This zone allows a user to remove the entity from the list.

Group

This type of migration request points to other migration requests. This allows you to define separate migration requests that represent logical groupings of migration plan instructions for ease of maintenance, but to combine all the separate migration requests into a single "grouped" migration request for streamlined export / import purposes.

The CMA export process will build an extract that includes the union of all the objects that qualify for the export and group them together based on their relationships.

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 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, a To Do Type includes a collection of valid To Do Roles and in turn the To Do Role refers to its To Do Types. If an implementation wants to copy a single To Do and include all its related information, including its To Do Roles, it does not want the migration of each To Do Role to in turn copy all its To Do 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. Also be sure that this FK reference defines an Information program.
- 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.

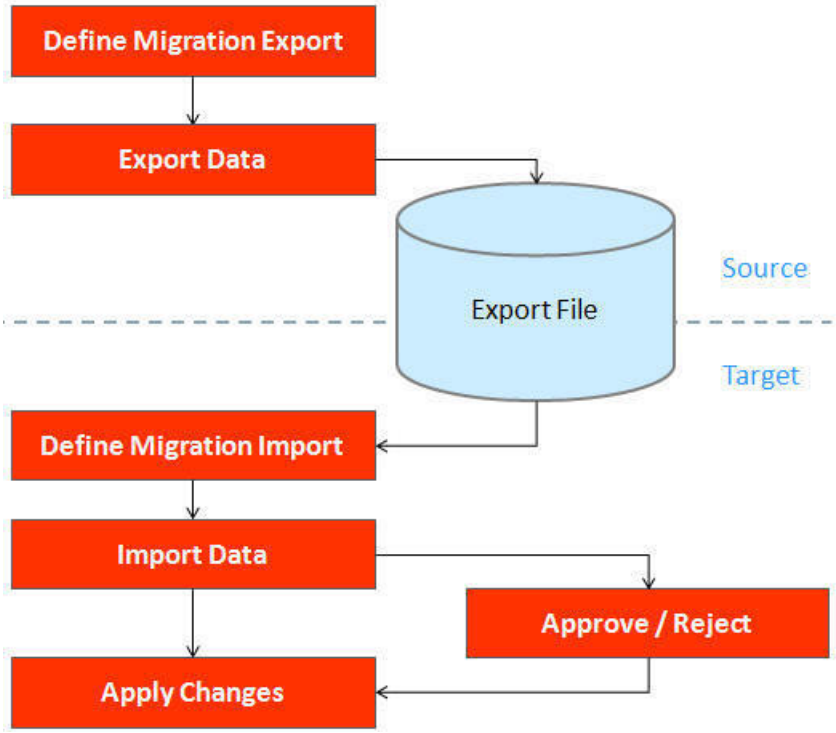
In order to support **Entity List** migration request, a default migration plan must be defined as an option on the maintenance object. This should be a single migration plan that supports all types of business objects for the MO.

If your implementation has a template migration request to use for 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

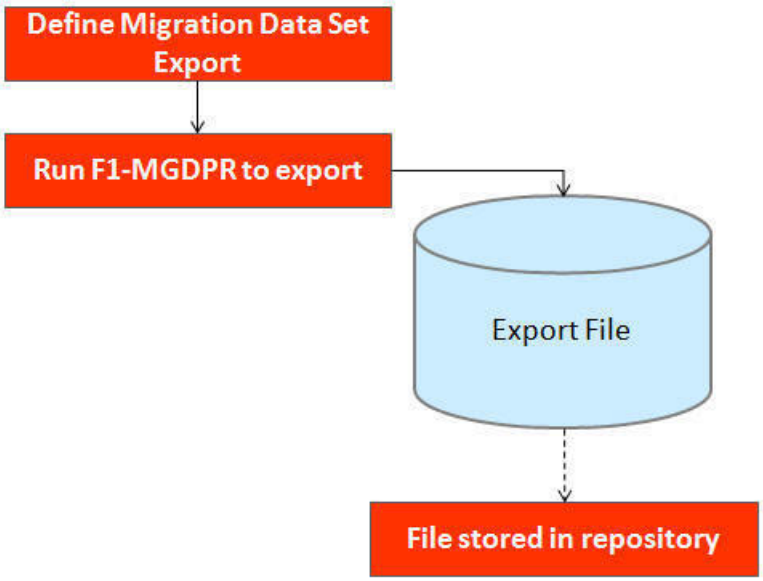
The following diagram illustrates a high-level view of the Configuration Migration Assistant execution process. The subprocesses illustrated here are described in more detail in the following sections.



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 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.



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.

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**.

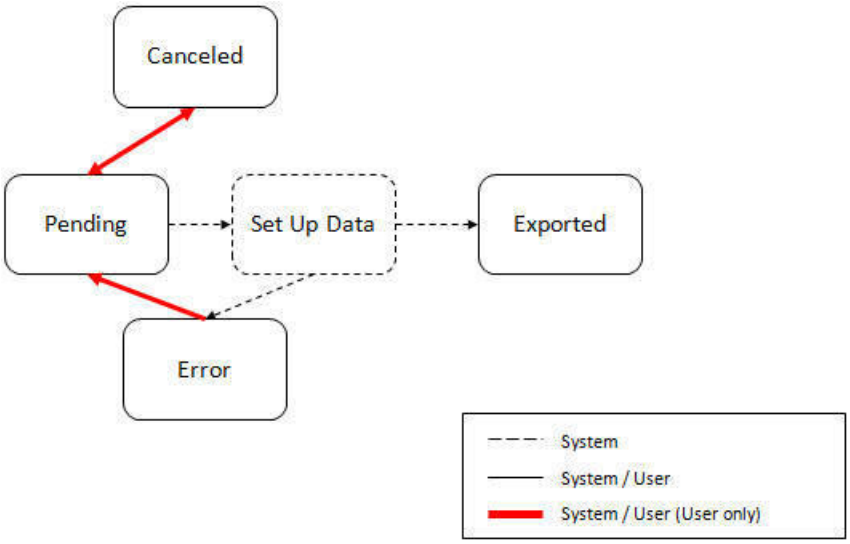
Enter a unique **File Name** for the export. Do not use spaces in the name, and do not enter the file extension or a path. The output location and file extension of the intended export file, which should appear in the **Export Directory** and **File Suffix** labels, are defined as described in the topic Migration Assistant Configuration.

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 content 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. Users may choose to suppress individual SQL statements for a given object that is approved. 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. Optionally, a migration import may be configured to automatically apply.
- **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 as described in the following topic Master Configuration - Migration Assistant 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.

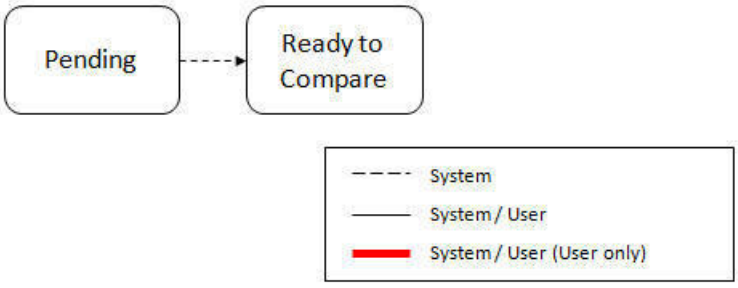
In addition, the user decides what the default status should be for resulting objects.

- 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 user may also configure the **Automatically Apply** flag to **Yes**. This allows for use cases where the migration is repetitive and has been tested and the user feels that there is no need for manual approval. Note that when configuring this setting, neither of the Default Status values may be set to **Needs Review** and at least one must be set to **Approved**.

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.



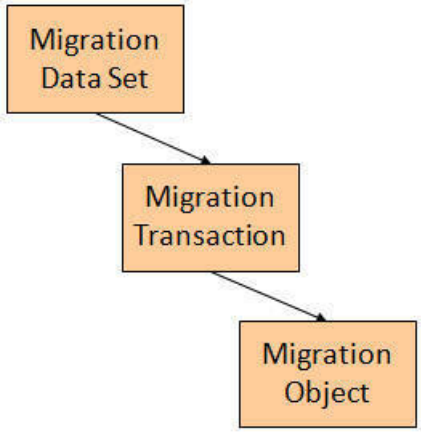
The following points describe the lifecycle.

- 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 and Running Batch Jobs 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.

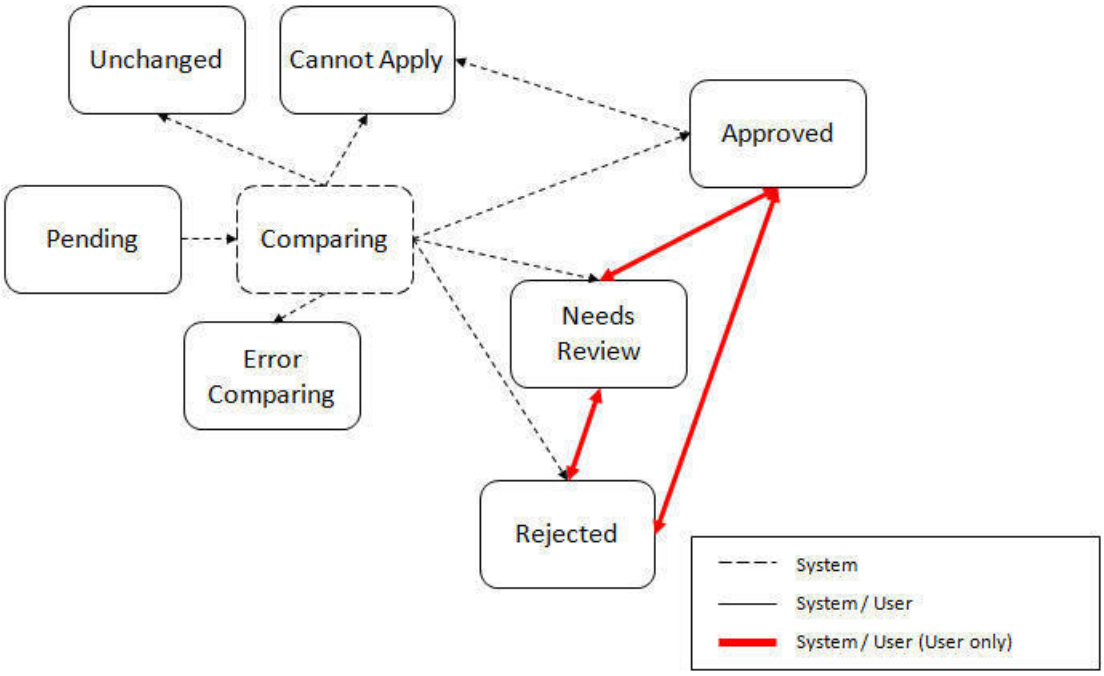
To cater for a possible large volume of objects, the comparison is done via a batch monitor. To aide in performance of the process, the monitor is performed on the migration objects so that it can be run multi-threaded. Once the objects are finished with the comparison, the migration transactions and 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.

NOTE: Refer to Running Batch Jobs for more information about streamlining the various steps in the process.

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.



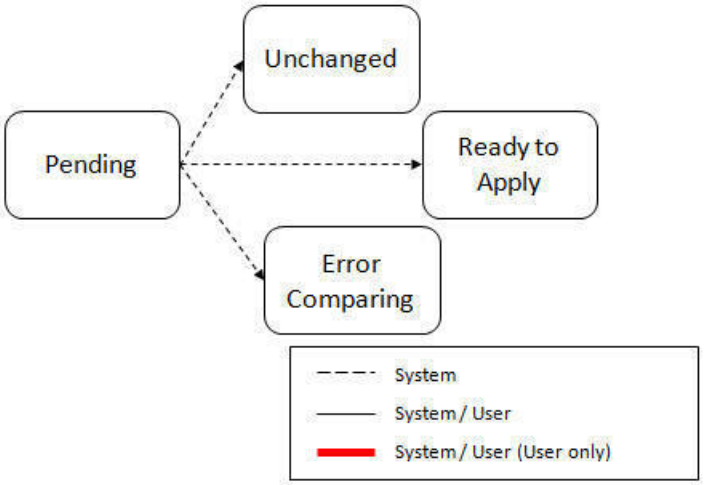
The following points describe the lifecycle.

- 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.

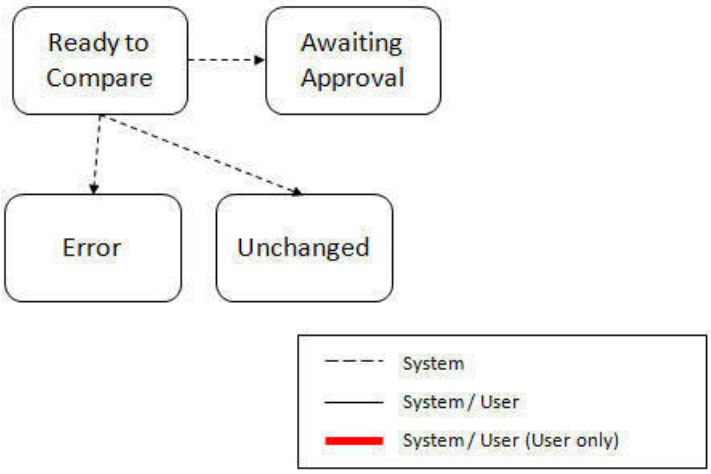
- If any related migration object is in the Error Comparing state, the transaction transitions to **Error Comparing**.
- If all related migration objects are in the Unchanged state, the transaction transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Ready to Apply**. This means 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.

A user has the option to mark one or more SQLs within a migration object to be suppressed. In this case, if the migration object is approved, then only the SQLs that are not marked as Suppressed will be applied. This option is useful when importing a record that has many child records. It may be that the user wants to only include a subset of the child records. This may be done by suppressing the child records that should not be included.

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.

Alternatively, if the Automatically Apply flag was set to **Yes** when creating the import record, the import data set will progress from **Awaiting Approval** to **Apply Objects**. Refer to Import Process Summary for more information.

> **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.

> **NOTE:** Refer to Running Batch Jobs for more information about streamlining the various steps in the process.

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 XML 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.

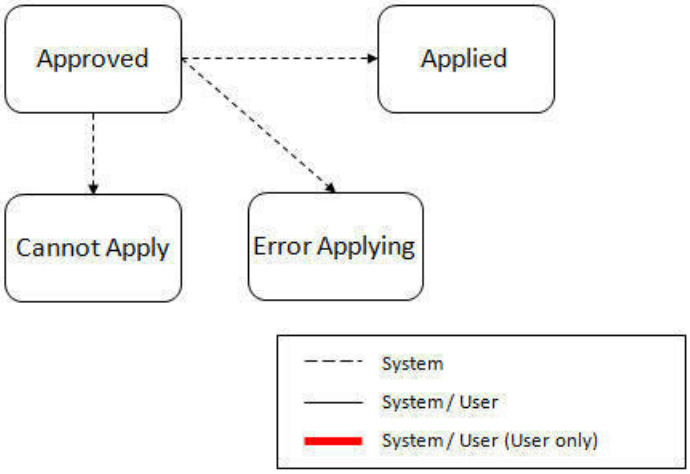
Apply Objects

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.

The following diagram is the portion of the migration object lifecycle that pertains to the Apply step.



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.

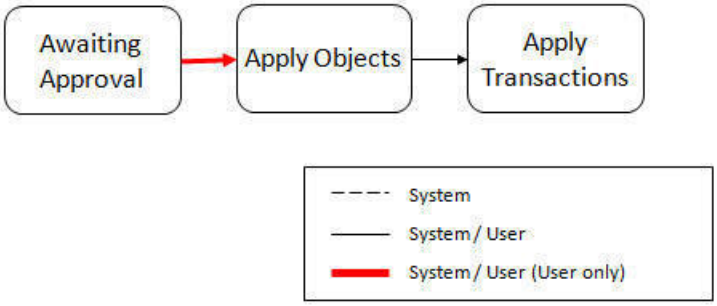
- They are in category 5 described above where the records have a circular reference with another record. For this scenario, the Apply Transactions step described below should successfully apply the records.
- There is some other error that is unrelated to the records in the current migration. In this case, manual intervention may be 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 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** or the count of records in **Error Applying** does not exceed a configured limit. If so, it automatically transitions the record to the **Apply Transactions** state.

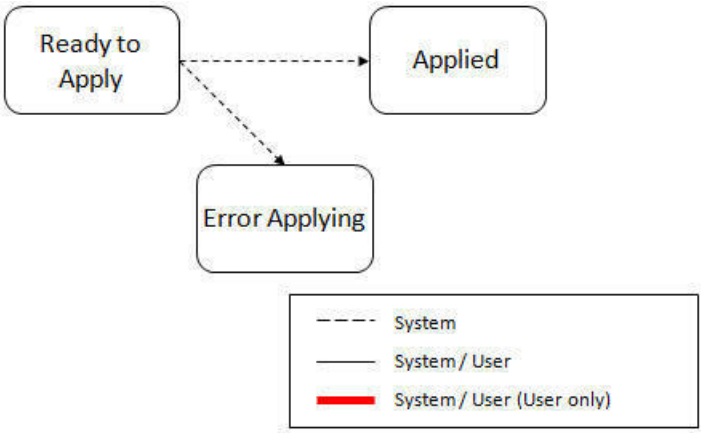
If the number of objects in **Error Applying** exceeds a configured limit, the monitor algorithm does not automatically transition the record. In that case, a user must determine if the large number of errors can be resolved or manually transition to **Apply Transactions** (despite the large number of errors). 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.



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. If the number of records in error is below a certain limit (and the data set monitor batch job is submitted to execute the monitor algorithms) the system will automatically transition the data set to the **Apply Transactions**. If the monitor batch job is not run or if the number of objects in error exceeds a certain limit, a user must make the decision after viewing the errors in the Objects in Error zone on the Migration Data Set Import portal.

- 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**. The following points describe the Retry logic for migration objects.

- If a user decides to **Retry Objects** (using an action button on the Migration Data Set Import page), the data set transitions to the **Retry Objects** state. At this point the Migration Object monitor must be run.
- The monitor on the **Error Applying** state for the objects detects that the data set is in the state of **Retry Objects** and that triggers the transition back to **Approved**.
- The next step is to transition the data set from **Retry Objects** to **Apply Objects**. This may be done manually or by running the Migration Data Set Import monitor process.
- Now the objects are eligible to be picked up by the object level Apply process.

Analogous logic exists for the migration transactions.

- If a user decides to **Retry Transactions** (using an action button on the Migration Data Set Import page), the data set transitions to the **Retry Transactions** state. At this point the Migration Transaction monitor must be run.
- The monitor on the **Error Applying** state for the transactions detects that the data set is in the state of **Retry Transactions** and that triggers the transition back to **Ready to Apply**.
- The next step is to transition the data set from **Retry Transactions** to **Apply Transactions**. This may be done manually or by running the Migration Data Set Import monitor process.
- Now the transactions are eligible to be picked up by the transaction level Apply process.

The retry logic may also occur when transitioning between the Apply Objects and Apply Transactions depending on whether or not there are errors. The following scenario highlights this point.

- After the **Apply Objects**step there are objects in **Error Applying**. The data set transitions to **Apply Transactions** and the Apply step is done at the transaction level.
- After the **Apply Transactions** step there are transactions in **Error Applying**.
- User chooses to try to apply objects again (by clicking **Retry Objects**). The steps outlined above for retrying objects are followed at this point.
- After the apply objects, user may choose to retry objects again (after fixing errors if applicable).
- At some point the user will transition to**Apply Transactions** again. If there are transactions in **Error Applying**, the system will automatically transition the data set to **Retry Transactions** and the steps outlined above for retry transactions are followed.

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 **Completed** 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
	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
	21	**Migration Transaction status update**	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply, Unchanged or Error Comparing	22
	22	**Migration Data Set Import status update**	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Awaiting Approval, Apply Objects (if configured for Automatic Apply), Unchanged or Error	30
Approval	**30**	**Review comparison results, approve / reject as needed** (This step is skipped if the data set is configured for Automatic Apply)	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.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
	41a	Migration Data Set Import status update - auto transition to Apply Transactions. Only applicable if the number of migration objects in Error Applying is below a threshold	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42
	41b	**Migration Data Set Import status update - manual transition to Apply Transactions.** Occurs if user reviews errors and determines that they may be resolved in the Apply Transaction step.	Manual	Migration Data Set Import - click Apply Transactions			
		System detects that all the transactions are in the Ready to Apply state and proceeds to the Apply Transactions state.				Migration Data Set Import - Apply Transactions	42
		System detects that there are transactions in the Error Applying state and transitions instead to the Retry Transactions state.				Migration Data Set Import - Retry Transactions	45
	41c	Migration Data Set Import status update - manual transition to Retry Objects. 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.	Manual	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Retry Objects	44

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
	42	**Apply Transactions**	Batch		F1-MGTAP - Migration Transaction Monitor - Apply	Migration Transaction - Applied or Error Applying Migration Object - Applied or Error Applying	43 Appropriate next action is based on error review, if applicable.
	43a	**Migration Data Set Import status update - auto transition to Completed** Applicable if all transactions are Applied	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Completed.	N/A
	43b	Migration Data Set Import status update - manual transition to Retry Objects. 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.	Manual	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Retry Objects	44
	43c	Migration Data Set Import status update - manual transition to Retry Transactions. 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.	Manual	Migration Data Set Import - click Retry Transactions		Migration Data Set Import - Retry Transactions	45
	44	Migration Objects status update from Error Applying back to Approved. Occurs if user chose to Retry Objects.	Batch		F1-MGOPR - Migration Object Monitor	Migration Object - Approved	
		Migration Data Set Import status update from Retry	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Objects	40

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		Objects to Apply Objects					
	45	Migration Transactions status update from Error Applying back to Ready to Apply. Occurs if user chose to Retry Transactions or if the user transitions to Apply Transactions and the system detects that there are Transactions in the Error Applying state.	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply	42
		Migration Data Set Import status update from Retry Objects to Apply Objects	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	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	Processes data set records in the following states: • Pending • Ready to Compare • Apply Objects • Retry Objects • Apply Transactions • Retry Transactions
F1-MGTPR	Migration Transaction Monitor (Deferred)	Processes transaction records in the following states: • Pending • Error Applying
F1-MGTAP	Migration Transaction Monitor - Apply	Processes transaction records in the Ready to Apply state where the data set is in the Apply Transactions 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.
F1-MGOPR	Migration Object Monitor	Processes object records in the following states: • Pending

Batch Control	Description	Comments
		• Needs Review (check for Data Set cancellation) • Rejected (check for Data Set cancellation) • Error Applying
F1-MGOAP	Migration Object Monitor - Apply	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, including ways to automate the above steps.

Cancelling 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.).
- If multiple migration requests are exported at the same time, on the import side, 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 the object to transition to "Cannot Apply". If instead you wait until the first file is completed 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. This may also be avoided by using a Group migration request to include all objects in one file rather than multiple files.
- The system provides an algorithm to purge "unchanged" migration objects for a given migration data set. This may be plugged in as a BO exit algorithm on the **Ready to Compare** state for the Migration Data Set Import business object (**F1-MigrDataSetImport**).

Caching Considerations

There are two important caching considerations when using CMA to migrate administrative data.

- The "apply" batch processes update data. As described in Batch Cache, if any batch job is updating data marked as "cached for batch" it must be run in a special thread pool. Contact your system administrator and ask for the name of

a thread pool that has turned off caching. This thread pool name must be entered in the batch controls used for CMA processing in the **Thread Pool Name** parameter.

- 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.

FASTPATH: Refer to Running Batch Jobs for more information about running CMA batch jobs.

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. A user may choose to update the status of one or more records by checking the records and clicking **Approve**, **Reject** or **Needs Review** accordingly.
- **Migration Data Set Objects in Error.** This zone is only visible if there are objects for this data set in a non-final status that have errors. 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 drill into the record to transition it to **Error Applying** or choose to manually fix the cause of the errors and click **Retry Objects**. The user may also choose to select one or more records to **Reject**.

  NOTE: Refer to Apply Step for more information about resolving errors.

- **Migration Data Set Transactions in Error.** This zone is only visible if there are transactions for this data set in a non-final state that have errors. It indicates the error for each transaction. 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. A user may choose to update the status of one or more records by checking the records and clicking **Approve**, **Reject** or **Needs Review** accordingly.

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.

The **Migration Object - List of SQL Statements** zone lists the SQL statements that have been generated as a result of the comparison step. A check mark in the Suppressed checkbox indicates that the SQL for that row will not be applied during the apply step.

The **Migration Object - Algorithms** zone displays any pre-compare algorithms that are associated with the migration plan for this migration object.

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 and batch control post processing plug-in algorithms that automatically submit the appropriate next batch job for that step in the process. This allows for as much automation as possible for the steps that don't require user input. Note that configuration is required because the BOs / batch controls are not configured for this scenario by default. The following table highlights the BO and status where an algorithm may be plugged in and the name of the algorithm to use.

Business Object	Status	Algorithm
Migration Data Set Export	**Pending**	**F1-MGDPR-SJ** (Submit Migration Data Set Export Monitor).
Migration Data Set Import	**Pending**	**F1-MGDIM-SJ** (Submit Migration Data Set Import Monitor).
	Ready To Compare, **Retry Objects**	**F1-MGOPR-SJ** (Submit Migration Object Monitor).
	Apply Objects	**F1-MGOAP-SJ** (Submit Migration Object Apply Monitor).
	Apply Transactions	**F1-MGTAP-SJ** (Submit Migration Transaction Apply Monitor).
	Retry Transactions	**F1-MGTPR-SJ** (Submit Migration Transaction Monitor).

The following table highlights the batch controls where an algorithm may be plugged in and the name of the algorithm to use.

Batch Control	Algorithm
F1-MGTPR (Migration Transaction Monitor)	**F1-MGDIM-NJ** (Submit Migration Data Set Import Monitor).
F1-MGTAP (Migration Transaction Monitor - Apply)	**F1-MGDIM-NJ** (Submit Migration Data Set Import Monitor).
F1-MGOAP (Migration Object Monitor - Apply)	**F1-MGDIM-NJ** (Submit Migration Data Set Import Monitor).
F1-MGOPR (Migration Object Monitor)	**F1-MGTPR-NJ** (Submit Migration Transaction 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 for the CMA batch processes used to migrate administrative data, there are multiple important caching considerations. Refer to Caching Considerations for more information.

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 wait until the supported number of threads are finished, possibly resulting in a large number of errors in the Apply steps.

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.

The following points highlight some information about Framework-provided migration requests. Navigate to the migration request page in the application to view the details of all provided objects.

- Several base migration requests are supplied to logically group system and administrative tables. For example, there is a migration request for Framework System Configuration **F1-SystemConfig** where most system configuration objects are included. There is another one provided for CMA related configuration objects.

- The system supplies a group migration request **F1–FrameworkConfig** (Framework Configuration), which includes several other migration requests. The expectation is that this migration request includes all the typical objects that are included in a wholesale migration. Your specific product may include this migration request into its own group migration request to support a wholesale migration of all the framework and product administrative tables. An implementation may choose to build a custom group migration request. In this case, review the various migration requests provided by base to see if any may be included as components for the custom migration request. Then any new migration plans added to the base migration request in future releases are automatically included in future migrations.
- There are several different security related migration requests that include different combinations of migration plans to support multiple possible business requirements related to security migration. Note that the security migration request included in the above mentioned group migration request is the one that does not include users. If your implementation wants to copy users to a target environment, refer to Importing Data that References a User for some considerations.

NOTE: 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 following points highlight some information about the Framework-provided migration plans. Navigate to the migration plan page in the application to view the details of all provided objects.

- Fields and characteristic types are not migrated with an object (like a business object or a data area) unless specifically indicated.
- The **Application Service** used by an object is migrated only if it is CM-owned.
- The **Batch Control** object optionally references a User (for 'timed' batches). Refer to Importing Data that References a User for some considerations about copying the user. 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, the system does not include these records 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.

- 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 targeted 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 targeted 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.
- For scripts, schema-based objects and zones, the migration plans provided by the product migrate, through constraints, some of the typical associated data 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. The iterative processing functionality of the import step should resolve any timing issues that may result in validation errors for these types of objects.
- 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). 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. It may be included in any appropriate custom targeted 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 (**Administrator** 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 Administrative Data with System Generated Primary Keys, 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 administrative data (common attachments) and non-administrative 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.
- For the system messages, the product provides three different migration plans.
  - Message Category and its Messages (F1-MessageCategory). This migration plan is included in the **F1-SystemConfig** migration request.
  - Message Category (F1-MessageCategoryOnly). This migration plan is provided to support a targeted migration where an implementation has created a custom message category and wants to move it but doesn’t want to move all its messages.
  - Message (F1-Message). This migration plan is provided to support a targeted migration where only specific messages within a message category should be migrated.
- For lookup values, the product provides two different migration plans.
  - Lookup Field and its Values (F1-Lookup). This migration plan is included in the **F1-SystemConfig** migration request.
  - Lookup Value (F1-LookupValue). This migration plan is provided to support a targeted migration where only specific lookup values within a lookup field should be migrated.
- 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-SystemConfig** 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**.
- Many of the integration related maintenance objects that include references to environment-specific data, such as Message Senders. This data should be migrated with extreme care. When appropriate, consider taking advantage of URI substitution. Refer to Referencing URIs for more information.

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.

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.
- If 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.

Conversion

This section describes the overall conversion process and the tools provided to support it.

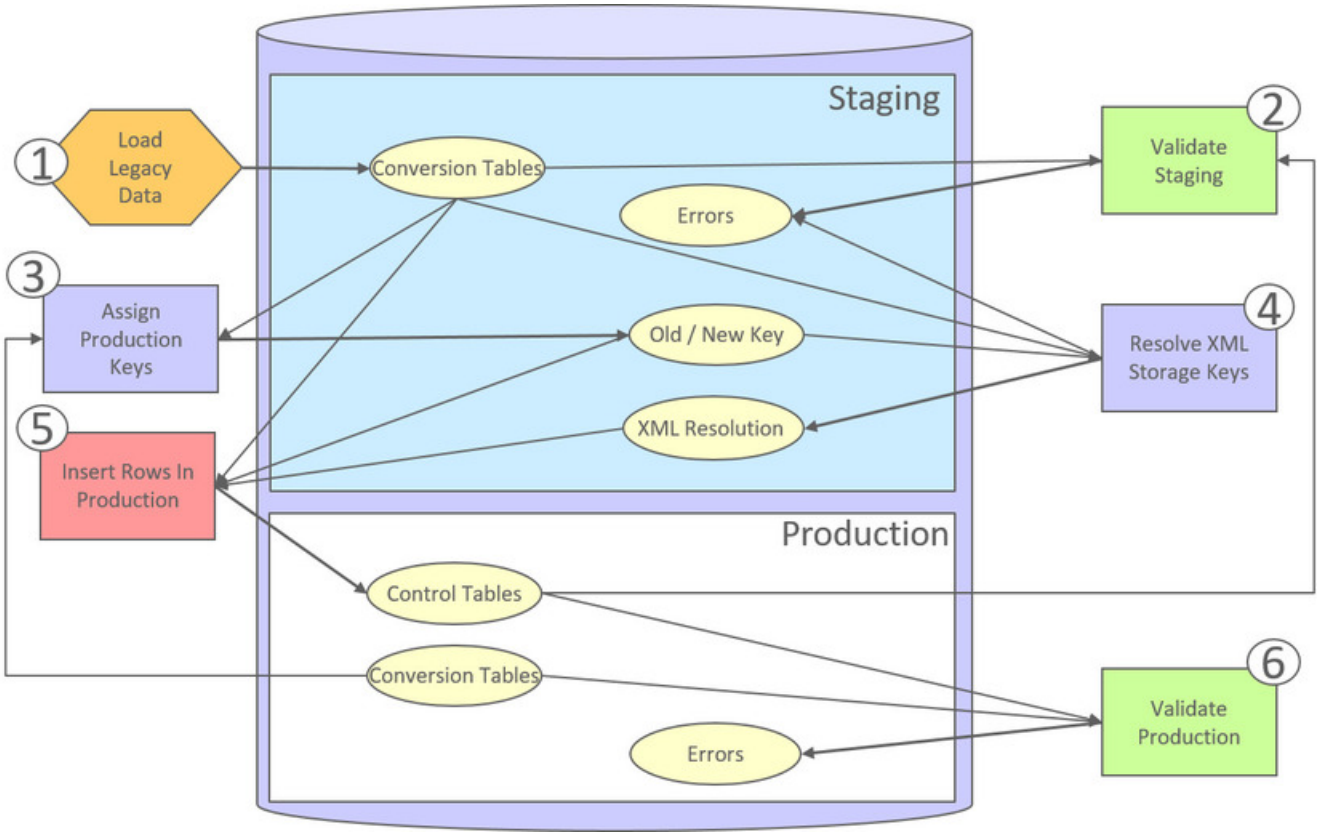
Understanding The Conversion Process

When you're ready to convert data from your legacy system, you will have analyzed your processing requirements according to your business and organizational needs and set up the control tables accordingly. After the control tables are set up, you are ready to load data into the system from your legacy system.

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. The production owner is linked to the tables used by your production system.

> **NOTE:** It's important to notice that **control tables exist only in production**. In staging, they exist as views to the production schema. Refer to Multiple Owners In A Single Database for more information about table ownership.

The conversion effort involves several steps as illustrated in the following diagram:



The following points briefly outline each of the above tasks:

- **Load Legacy Data.** During this step, your legacy data is loaded 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. 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 Staging.** During this step, the system validates the data you loaded into the staging tables. Note that validation rules validate your staging data using the control tables that have been set up in production.
- **Assign Production Keys.** During this step, the system allocates random, clustered keys to the rows in the staging database.
- **XML Resolution.** During this step, the system resolves legacy keys that may be mapped to an XML storage field with new values that were assigned in the key assignment step.
- **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 and foreign keys are reassigned using the data populated in the key assignment and XML resolution steps.
- **Validate Production.** During this step, you rerun the object validation processes, but this time against production. We recommend performing this step to confirm that the insertion programs have executed successfully.

NOTE: Additional steps may be needed to support specific data conversion requirements. Refer to your product's documentation for additional conversion related information.

This section further discusses concepts and guidelines related to the conversion process.

Conversion Entities

Conversion is enabled for selected master and transaction maintenance objects. In the same way, only a subset of a maintenance object's tables are enabled for conversion.

The following is provided for each maintenance object that supports conversion:

- An entity validation batch control. This batch process validates data in staging and can also be run to validate data in production.
- A key assignment batch control for each **table** that is eligible for conversion and has a system-generated key.
- An XML resolution batch control if the maintenance object has at least one table that includes an XML storage field.
- An insertion to production batch control for every **table** of the maintenance object that is eligible for conversion.

You may review the conversion configuration provided for an entity and more using the Conversion Entity Dashboard portal.

Conversion Steps

The following sections provide more details about the steps in the conversion process.

NOTE: Additional steps may be needed to support specific data conversion requirements. Refer to your product's documentation for additional conversion related information.

Load Legacy Data Into Staging Tables

This section provides some high level discussion about mapping and loading legacy data to the system's staging tables.

Mass Load Utility

Unless in a Cloud installation, you can use any method you prefer to load 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.

NOTE: In a Cloud installation, refer to the "Data Conversion Support for Cloud Implementations" for more information about the various tools provided to support database related conversion tasks.

Populating Keys

Some tables in the staging database have keys that are system-assigned random numbers. Special consideration should be given when populating these keys. Refer to A Note About Keys for more information on how to properly populate these tables. 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.

Populating Characteristic Tables

There are many maintenance objects that include a characteristic table used to capture miscellaneous information about the object. 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.

Validate Information In The Staging Tables

During the first validation step, the system validates the data you loaded into the staging tables. This section provides some high level discussion about the validation process.

A Batch Process Per Maintenance Object

Each of the maintenance objects that are eligible for conversion must be validated using its respective entity validation conversion batch process. These are classic background processes that can also be run against production data.

These processes are multi-threaded and have no dependency on other processes.

Validation Mode

By default, each entity is fully validated for schema as well as entity validation rules, at the maintenance object and business object level. The batch process performs the same business rules that are used to validate data added by users in your production system. You may restrict the batch process to perform schema validation only or entity validation rules only using a batch parameter. This may be useful in identifying common data issues in preliminary runs.

Random Sample Mode

By default all records are selected for validation. The "Process Every Nth" batch parameter allows you execute this process in a random-sample mode to highlight pervasive errors. In this mode, you are actually telling the batch process to validate every N records.

We strongly recommend validating each entity in the following steps:

- Execute the validation batch process in random-sample mode.
- Review and correct the errors. Note, you can use the base package's transactions 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 batch process in all-instances mode to highlight elusive, one-off errors.

Reported Errors

Errors encountered during validation are logged onto the conversion Validation Error (CI_VAL_ERR) table. Note that at the start of this job, all rows in the conversion error table for the process maintenance object are deleted.

You can view errors highlighted by the validation process using the Conversion Entity Dashboard portal.

Recommendations To Speed Up Validation

The following points describe ways to accelerate the execution of the validation process:

- Ensure that statistics are recalculated after data has been inserted into the staging tables.
- Execute the process 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.

- Remember that validation can be run in random-sample mode. We recommend running these batch processes using a large sample value for this parameter until the pervasive problems have been rectified.

NOTE: In a Cloud installation, refer to the “Data Conversion Support for Cloud Implementations” for more information about the various tools provided to support database related conversion tasks.

Another use for these programs

In addition to validating your objects after conversion or an upgrade, the validation programs have another use. Say for example, you want to experiment with changing the validation of a business entity and you want to determine the impact of this new validation on your existing records. You could change the validation and then run that entity’s validation batch process- it will produce errors for each record that fails the new validation.

Allocate Production Keys

During the key assignment step, the system assigns a new key for each legacy key for a table that its prime-key requires a system-generated random key. 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).

The topics in this section provides some high level discussion about the key assignment process and describe the background processes used to assign production keys to the staging data.

The Old Key / New Key Table

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 convention "<1st letter of owner flag>K_<table_name>" is used to denote the old key / new key table name. For example, the old key / new key table for CI_ACCT is called CK_ACCT.

The insertion batch process that transfers 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. In the same way, the XML resolution process resolves conversion foreign keys residing in XML storage fields and replaces them with their corresponding new keys from these table.

A Batch Process Per Table

A key assignment batch process is provided for each table that has a system-generated key and belongs to a maintenance objects that is eligible for conversion. The batch process is responsible for populating the corresponding old key / new key table (i.e., you don't have to populate these tables). These processes are single threaded. Refer to a table’s corresponding key assignment batch process for more information.

Key Assignment Dependencies

Most tables with system-generated keys do not inherit part of their key from another table’s key. Their corresponding key assignment batch process have no dependencies and can therefore be executed in any order you please.

Some tables inherit part of their key from another table’s key. Key assignment batch processes for such tables must be executed in key inheritance order. In other words, the key assignment process for a table should be run after the process that generates keys for the entity it depend on. The Conversion Entity Dashboard portal displays key inheritance dependency information if any exist for a conversion entity.

NOTE:

You may run multiple key assignment batch process in parallel as long as they are independent with respect to key inheritance.

Iterative Conversions

Rather than perform a "big bang" conversion (one where all records of an entity are populated at once), some implementations have the opportunity to go live on subsets of their entity 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 and unique keys.

Key assignment logic creates the initial values of keys by a manipulation of the sequential 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 key assignment batch process allows you to specify a starting value for that row number. 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. This parameter is only used if you are performing conversions where data already exists in the tables in the production database.

Run Type

Key assignment is performed in two steps:

- **Initial Key Generation.** The system allocates new keys to the rows in the staging tables (i.e., it populates the respective old key / new key table).
- **Duplicate Key Resolution.** The system reassigns keys that are duplicates compared to production.

By default, both steps are performed at the same run but you have the option to run them separately by indicating which step to run via a batch parameter. The proper use of this parameter will greatly speed up the key assignment step as described under **Recommendations To Speed Up Key Generation** section.

Recommendations To Speed Up Key Generation

The following points describe ways to accelerate the execution of the key generation programs.

- For a non-Cloud installation:
  - 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 old key / new key 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 inheritance dependency 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.
- Optimal use of the **Run Type** batch parameter.
  - Before any key assignments, alter both the "old key" F1_CX_ID index and the "new key" CI_ID index on the old key / new key tables to be unusable.

- Run all key assignment batch processes in tiers, submitting each job to only perform the **Initial Key Generation** step.
- Rebuild the indexes on the old key / new key tables. Rebuilding the indexes using both the PARALLEL and NOLOGGING parameters will speed the index creation process. Statistics should be computed for these indexes.
- Run all key assignment batch processes in the current tier that were previously run in initial key generation mode, to perform the **Duplicate Key Resolution** step. This will reassign all duplicate keys.

NOTE: In a Cloud installation, refer to the "Data Conversion Support for Cloud Implementations" for more information about the various tools provided to support database related conversion tasks.

XML Resolution

Most system-generated foreign keys are stored in physical fields and characteristic tables and as such their legacy value is replaced with its corresponding new key as part of the insertion to production process. Rarely, maintenance objects may store system-generated foreign keys in an XML storage field, i.e. a field defined with the data type of CLOB or XML. This step is only applicable to such maintenance objects.

During this step, the system resolves convertible system-generated foreign keys that may reside in XML storage fields you may have loaded into the staging tables. This section provides some high level discussion about the XML resolution process.

The XML Resolution Table

It's important to understand that the system does not overwrite the prime-keys on the XML storage fields in the staging database, as this is a very expensive IO transaction. Rather, a corresponding XML resolution table exists for each table that defines an XML resolution field to capture each row's resolved XML storage content, i.e. the content with all the old keys replaced with the new assigned keys.

The convention "<1st letter of owner flag>R_<table_name>" is used to denote the XML resolution table name.

The insertion batch process that transfers the rows into the production database replaces each XML storage field with its resolved value from the corresponding XML resolution table.

A Batch Process Per Maintenance Object

Each of the maintenance objects that are eligible for conversion and support XML storage fields is provided with an XML resolution batch process. These batch processes must be run to resolves foreign keys that may reside in these XML storage fields.

These processes are multi-threaded and must be executed after the key assignment step has completed and before inserting data to production.

XML Resolution Eligibility

Not all maintenance objects that support XML storage fields actually store convertible system-generated foreign keys in their XML storage field. If none of the business objects associated with the maintenance object involve mapping of such foreign keys to an XML storage field then XML resolution is not needed for any row in the maintenance object. The XML resolution batch process detects such situation and completes right away without storing any rows in any of the maintenance object's XML resolution tables.

Only Resolved Values Are Captured

XML storage fields typically store large amounts of data. To avoid capturing the same XML content redundantly, the system only stores values in the resolved XML storage fields if the resolved value is different than the original value, i.e. at least one key was resolved.

If for a given record the resolved XML content is the same as the original content then the following rules apply:

- If this is the primary table of the maintenance object a record is inserted into its corresponding XML resolution table for that record with no value in the XML storage field.
- If this is a child table of the maintenance object then no record is inserted into the corresponding XML resolution for that record.

Reported Errors

Errors encountered during XML resolution are logged onto the conversion Validation Error (CI_VAL_ERR) table. Note that at the start of this job, all rows in the conversion error table for the process maintenance object are deleted.

You can view errors highlighted by the XML resolution process using the "Validation Error Summary" page.

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.

A Batch Process Per Table

An insert to production batch process is provided for each table of a maintenance objects that is eligible for conversion.

The batch process is responsible for transferring to production all the rows from the staging table while replacing all references to old keys with their corresponding new keys as follows:

- Keys residing in prime keys, foreign key fields and characteristics are resolved using the corresponding old key / new key tables.
- Keys residing in XML storage fields are resolved using the corresponding XML resolution tables.

All insertion batch processes are independent and may run concurrently. Also note, all insertion batch processes can be run in many parallel threads as described in the next section (in order to speed execution).

Recommendations To Speed Up Insertion

The following points describe ways to accelerate the execution of the insertion batch processes:

- Before running the first insertion batch process:
  - Rebuild the index on the prime key on the old key / new key table.
  - Re-analyze the statistics on the old key / new key table.
  - 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.

NOTE: In a Cloud installation, refer to the "Data Conversion Support for Cloud Implementations" for more information about the various tools provided to support database related conversion tasks.

Validate Production

During this step, you rerun the object validation batch processes, but this time in production. We recommend rerunning these batch processes to confirm that the insertion batch processes have executed successfully. We recommend running these batches 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 processes in full validation mode (to validate every object).

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 key is allocated, the key assignment program checks the corresponding 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 a table whose prime key is a system-assigned is defined on its table definition record.
- 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.

Multiple Owners In A Single Database

The conversion process relies on the following table owner configuration in the system database:

- The production owner is linked to the tables used by your production system. These tables have an owner ID of CISADM.
- The staging owner is linked to the tables into which you insert your pre-validated data. These tables have an owner ID of CISSTG.

The staging owner schema is almost identical to the production schema, with the following exceptions:

- Control tables are not actual tables in staging but rather views to the corresponding tables in production.
- Conversion specific tables designed solely to support the conversion process exist only in the staging schema. For example, tables that manage key assignment and XML resolution.

This section provides some high level concepts related to these table owners.

Validation Always Uses Production Control Data

When the validation batch processes 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 entities needs to use the staging owner whereas the SQL statements that access control tables need to use the production owner. But notice that when these same validation batch processes run against production, 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.

  NOTE: In a Cloud installation, the application server may only point to a specific database user ID at any point in time. Refer to the “Data Conversion Support for Cloud Implementations” for more information about switching owners.

- The database user ID associated with the staging owner 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 owner uses CISADM as the owner for all 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.

Only Validation Can Work In Both Owners

While the redirection of owner ID's is a useful technique for the validation batch processes, it cannot be used by the key assignment and production insert batch processes? Why, because these processes have to access the same tables but with different owners at the same time. They also need to reference conversion specific tables that do not exist in production. For example, the batch process that inserts rows into a table in production must select rows from the staging version of that table, resolve keys from the conversion old key / new key mapping tables and insert the resolved records into the production version of that same table

This is accomplished as follows:

- In staging, a view to production exists for each eligible to conversion table. These views have hard-coded the database owner to point to 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.

Conversion Entity Dashboard

This portal provides an overview of a maintenance object’s conversion configuration along with information related to its current status and progress within the overall conversion process.

Refer to Understanding Conversion for more information.

You can access the portal from the **Admin Menu** > **Implementation Tools** > **Conversion Entity Query.** You are brought to a query portal with options for searching for a specific maintenance object that is eligible for conversion. Once a

maintenance object has been selected you are brought to the conversion entity dashboard portal to review the information for the selected record.

The following zones may appear as part of the portal's **Main** tab page.

- **Tables.** This zone provides information about tables that belong to the maintenance object and are marked as eligible for conversion.
- **Batch Controls.** This zone lists the conversion batch controls defined for the maintenance object in the order they should be executed within the conversion process along with current execution details if any.
- **Error Summary.** This zone provides a summary of all conversion errors related to the maintenance object.
- **Duplicate Keys.** This zone lists duplicate legacy keys that need to be resolved by the key assignment conversion process.

The **Errors** tab page provides a query zone that allows you to review conversion errors related to the maintenance object by various criteria.

Data Export

This section describes concepts and tools related to the various types of data export methods supported by the product.

Understanding Data Export

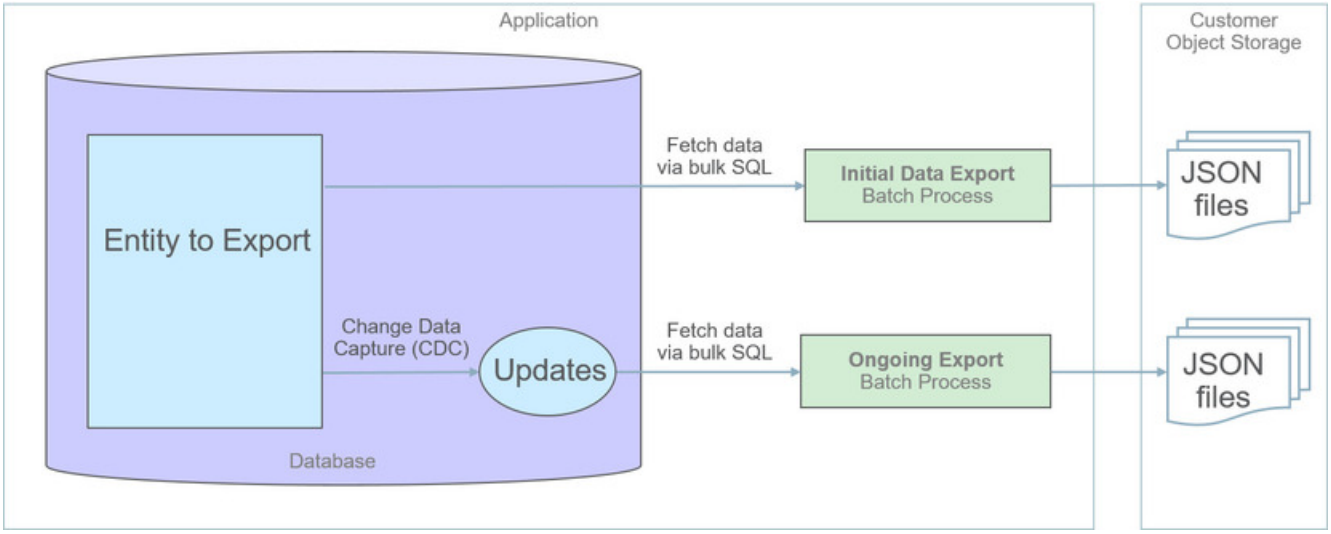
The application supports the following classes of export methods for a maintenance object:

- **Generalized.** This method may be used for maintenance objects that are low to medium in volume and update activity. Most maintenance objects are eligible for this method.
- **Specialized.** This method should be used for maintenance objects that are high in volume or very frequently updated.

This section discusses concepts and guidelines related to data export features.

Generalized Data Export

The generalized data export method provides a file-based export of data in JSON format. Exporting the data of a maintenance object using this method involves an initial export of the entity's entire data followed by an ongoing export of changes as they are made over time. These processes are illustrated in the following diagram:



The overall process involves the following steps:

- Enable entities for export. This step also allows you to enable ongoing export of these entity.
- Run the initial export initiator batch. This step submits a batch for each entity you have enabled for export.
- Schedule the ongoing export batch process to run periodically.

The following sections further discusses concepts and guidelines related to the generalized data export method.

Eligibility

By default a maintenance object is eligible for the generalized data export method unless explicitly marked otherwise by the **Data Export Class** option. The option may be used to mark an entity as either not allowed for any type of export or allowed for a specialized export only. The lack of such explicit option means the maintenance object is eligible for the generalized export.

One Method Fits All

The generalized data export method is designed to accommodate many eligible maintenance objects in a generic way that must also takes performance considerations into account. For this reason this method does not support entity specific filtering options nor custom rules of any kind.

The following rules apply to all entities that may be exported using this method:

- Entire data is exported. All fields on all tables that belong to the maintenance object are exported except for “key” and “log” tables if any.
- Same data structure and format is used for all entities. Refer to the **Record Format** section for more information.
- Data is exported to files only. Refer to the **Export To Files Only** section for more information.
- All changes are captured. It is not possible to exclude some changes from being tracked for ongoing export based on custom business rules.
- When data changes, a full snapshot of the data related to that entity is exported to avoid heavy data merging on the target side.

NOTE: Export is not supported for entities that rely on the logical time zone functionality. This type of definition is not commonly used and includes maintenance objects that at least one of their fields is defined as stored in **Logical Standard Time**.

Export To Files Only

The immediate destination of both initial and ongoing export processes are files in a customer specified location. In a Cloud installation files are created in a customer owned Object Storage.

Based on your business requirements you may further consume these files by downstream applications such as data lakes etc.

Enabling Data Export

To enable data export for an eligible maintenance object a **Data Export Control** record needs to be created. The record keeps an indication of whether the initial export for the maintenance object has been completed or not and whether changes made to the entity should be tracked and exported on an ongoing basis. A designated portal allows you to enable the generalized data export for many maintenance objects at once as well as monitor their current export status at a glance. Refer to Generalized Export Dashboard portal for more information.

NOTE: When applicable, ongoing data export should be enabled before submitting the initial export batch process for the entity. This ensures that changes made during the initial export are properly tracked.

Initial Data Export

Each maintenance object that is eligible for the generalized data export method is associated with an initial data export batch control that is referenced on the maintenance object using the **Export Batch Control** option. The same common batch program is used by all these batch controls to export the entire data of an entity to files. The process is multi-threaded and a separate file is produced for each thread.

Instead of manually submitting the generalized initial export batch process for each maintenance object you have enabled for export, submit the**F1-GEXPI** Generalized Initial Export Initiator batch process to automatically submit them all at once.

The initial export process for each maintenance object updates the corresponding data export control record for the maintenance object to indicate that the process has started and updates it again upon completion. This allows you to view the overall status of the initial export across all enabled entities on the Generalized Export Dashboard portal.

Note that the initiator batch process only submits batch processes for enabled maintenance objects that according to their data export control record their initial export has not started yet. If you need to export one or more entities again, reset the initial export status indication on their data export control records and submit the initiator batch process again.

The initial export file name is constructed as follows:

```
INIT_EXPORT[_file prefix(optional)]_[maintenance object]_[batch run]_[thread]_[thread count]_[timestamp].json
```

Refer to any of the individual generalized initial export batch controls and the initiator batch control for more information.

Ongoing Data Export

Ongoing data export is only supported for Cloud installations.

Data changes are tracked by the Capture Change for Ongoing Export (**F1-MO-REGCHN**) Maintenance Object Audit algorithm. The algorithm records the prime key of a changed entity in a designated **Data Export Update** table that serves as a queue of changed entities to export. The queue is later on consumed by the ongoing data export batch process.

> **NOTE:** The Maintenance Object Audit plug-in spot is capable of tracking changes made to the entity by any layer of the application. Making changes using direct SQL would bypass this mechanism and therefore should be avoided.

The audit rule is added to a maintenance object when the ongoing data export option on its corresponding data export control record is enabled. In the same way the audit rule is removed from the maintenance object when the ongoing export option is disabled.

> **NOTE:** The ongoing export option has to be enabled on the data export control record for the entity before it is initially exported. This ensures changes made while the initial export process is in progress are not missed.

Unlike the initial export batch process that requires a batch control to be defined for each eligible maintenance object, a single batch process is used to export ongoing changes made to all entities. The Generalized Ongoing Export (**F1-GEEXO**) batch control exports all queued up changes found in the Data Export Update table to a designated ongoing export file. A full snapshot of the data related to the changed entity is exported to avoid heavy data merging on the target side. The process is multi-threaded and a separate file is produced for each thread. Once exported, records are deleted from the Data Export Update table.

> **NOTE:** It is important to schedule the ongoing export batch process to run frequently enough to avoid accumulating a high volume of records in the Data Export Update table. The smaller the table is kept the faster it is for the change data capture mechanism to add records to it.

The ongoing export file name is constructed as follows:

```
INC_EXPORT_[batch run]_[thread]_[thread count]_[timestamp].json.gz
```

Record Format

Data is fetched and exported directly from the database in bulk read operations for performance reasons. For that reason, the structure used does not reflect the logical data model of the entity but rather its physical list of tables. Data for a given instance is organized by tables followed by rows in each table and for each row its list of fields in JSON format.

The following describes the format and structure used:

```
{
"OBJ": "<mo name>",
"TIMESTAMP": "<export time in ISO format for example 2019-07-25T11:06:04.740615Z>",
"PK1": "<mo pk1 value>",
"PK2": "<mo pk2 value if any>",  # PK2-5 should only be included when applicable
"PK3": "<mo pk3 value if any>",
"PK4": "<mo pk4 value if any>",
"PK5": "<mo pk5 value if any>",
"DELETED": true,  # should only be included when the entity is deleted
"DATA":
{
"<MO table name 1>":
[
{<name value pairs of all fields in row 1 in that table>},
{<name value pairs of all fields in row 2 in that table>},...
{<name value pairs of all fields in row n in that table>},...
],
"<MO table name 2>":
[
{<name value pairs of all fields in row 1 in that table>},
{<name value pairs of all fields in row 2 in that table>},...
{<name value pairs of all fields in row n in that table>}
],...
 "<MO table name n>":
[
 {<name value pairs of all fields in row 1 in that table>},
{<name value pairs of all fields in row 2 in that table>},...
{<name value pairs of all fields in row n in that table>}
]
}
```

Notes about field values and formats:

- All fields for a row are included even when the field is empty or null.
- All string values are trimmed of excess trailing spaces but if a field is empty it exported as an empty string "" and not null.
- An empty array is used for a table node when no records exists for the entity in the table.
- Date/time information is converted to UTC time zone and exported in ISO format for example 2019-07-25T11:06:04.740615Z.
- Deletes have an explicit indicator in the header.

Order of Consuming Files

It is important to apply all the initial export files for a maintenance object before applying ongoing change files to ensure data consistency.

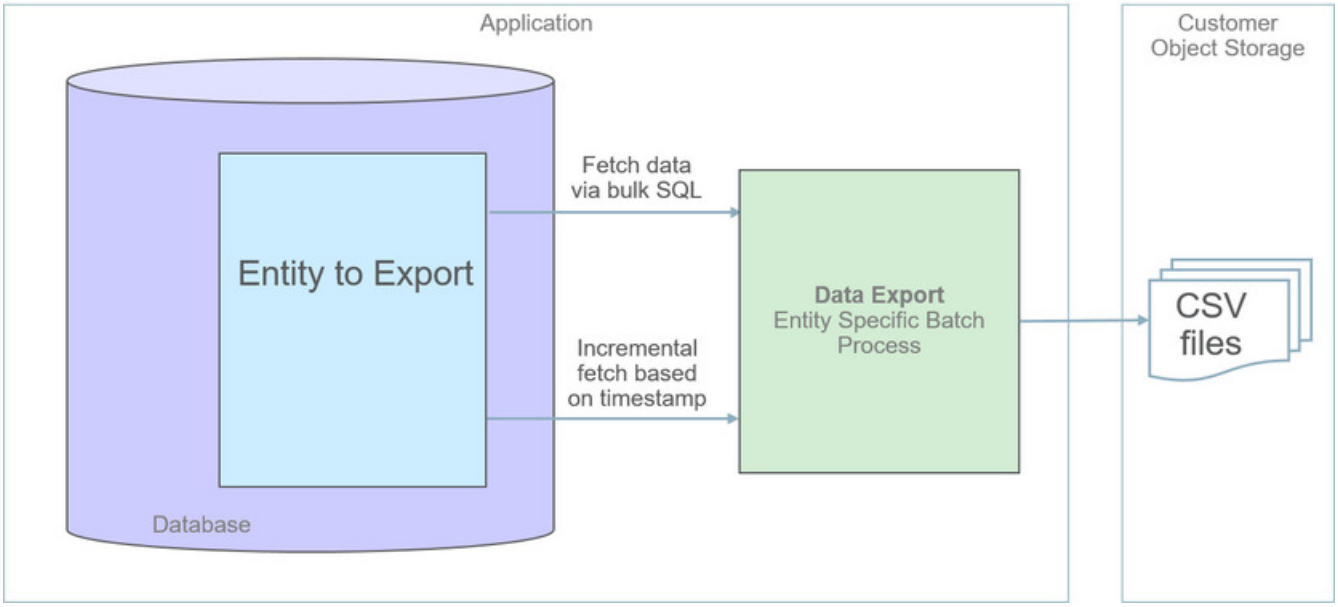
The following is a recommended procedure for consuming generalized data export files:

- Apply all initial data export files before ongoing export files.
- Apply initial files for a maintenance object in run number order.
- Apply ongoing export files in run number order.
- If a new initial export is available the file consumption procedure should be changed temporarily as follows:
  - Stop consumption of ongoing export files.

- Apply the new initial export files.
- Resume consumption of ongoing export files.

Specialized Data Export

Several performance considerations are called for when exporting the data of a high volume entity or an entity that is updated in extremely high frequency. The generalized method does not fit such entities and a specific batch process is designed for each, aiming to achieve an optimal export method for the entity. The initial run of the process exports the entity's existing data while subsequent runs of the same process exports data incrementally, i.e. it exports records that were added or updated since the previous run based a timestamp field. The overall process is illustrated in the following diagram:



The following sections further discusses concepts and guidelines related to the specialized data export method.

A Specific Process For Each Entity

A maintenance object that is designed to be exported using the specialized method is explicitly marked as such by the **Data Export Class** option and its specific batch process is defined using the **Export Batch Control** option.

All specialized batch processes are designed to export the data in a CSV format as it produces smaller files in size compared to more verbose formats. Refer to each batch process for more information about specific export considerations and parameters.

Enabling Data Export

To enable data export for a maintenance object a specialized type of Data Export Control record needs to be created. The record serves as an indication that such entity should be exported and keeps track of the date and time through which data has been exported so far, allowing subsequent export run to only export incremental changes.

Incremental Data Export

The specialized export method tracks changes incrementally from prior export run and does not use a change data capture mechanism. It therefore relies on a timestamp field on the primary table of the maintenance object to indicate when data was added or changed. Only entities that were added or changed since the previous export run are exported with each subsequent run. The date and time through which data has been exported is recorded for each maintenance object on its corresponding Data Export Control record.

Order of Consuming Files

Specialized data export files should be consumed in run number order to ensure a consistent and chronological uptake of data.

Defining Data Export Control

This portal is used to maintain Data Export Control records. These records capture your organization's export settings and current export state for a selected maintenance object. For a generalized export it keeps an indication as to whether the initial export for the maintenance object has been completed or not and whether changes made to the maintenance should be tracked and exported on an ongoing basis. For a specialized data export it keeps the date/time through which data was exported.

Refer to Understanding Data Export for more information.

You can access the portal from the**Admin Menu** > **Implementation Tools** > **Data Export Control.** You are brought to a query portal with options for searching for a specific record. Once a data export control record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The **Data Export Control** zone provides information about the selected record.

Generalized Export Dashboard

The number of maintenance objects that are eligible for generalized export is large. This portal allows you to easily select and enable export for many eligible entities at once as well as review the current export state of those already enabled at a glance.

Refer to Generalized Data Export for more information about the generalized data export method.

You can access the portal from the **Admin Menu** > **Implementation Tools** > **Generalized Export Dashboard.**

The following zones may appear as part of the portal's **Main** tab page.

- **Export Entities.** This zone lists entities for which generalized data export control settings have already been set.
- **Eligible Entities.** This zone lists entities that are eligible for the generalized data export method and for which such settings have not yet been set.

Chapter 2

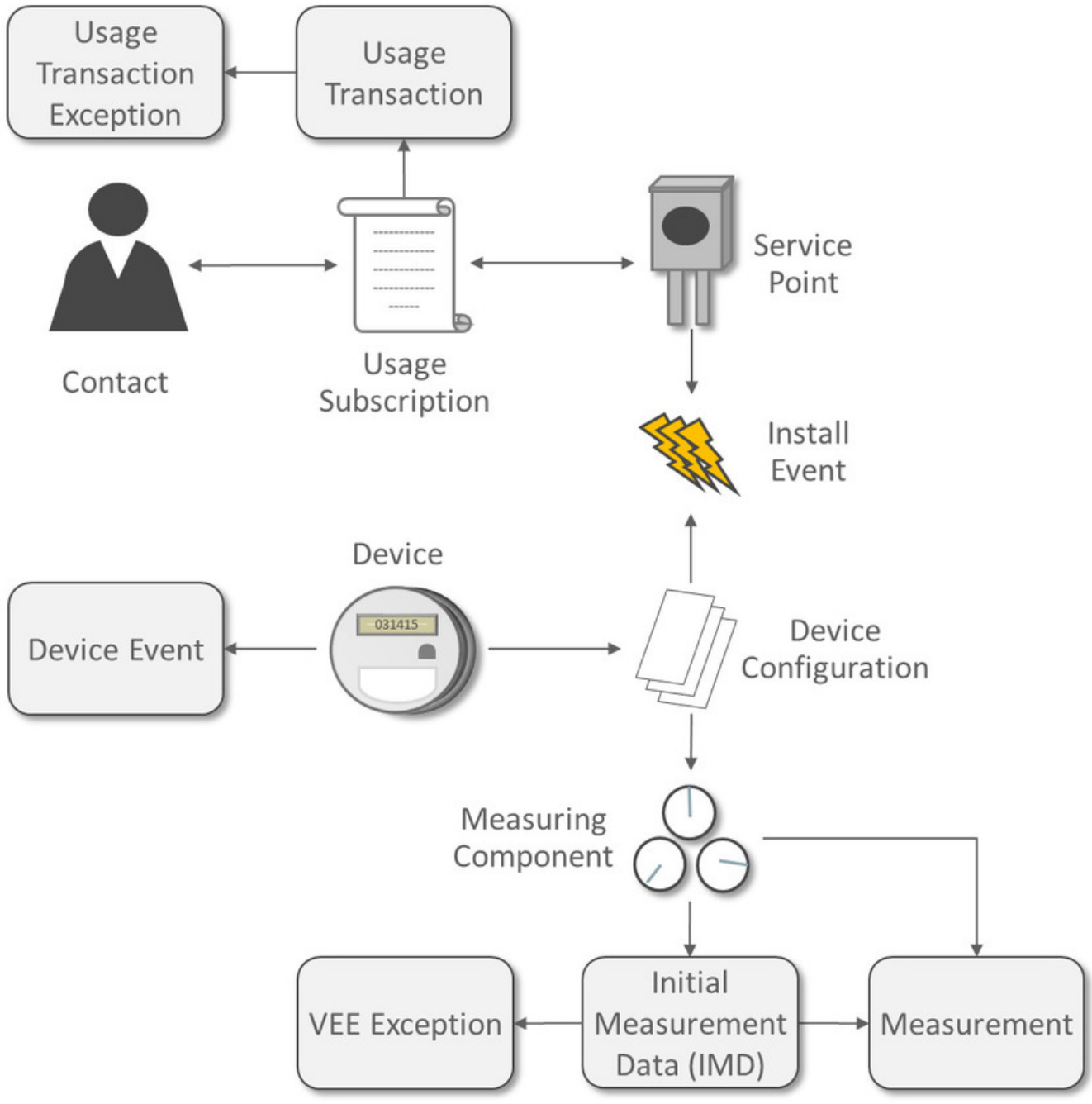
Meter Solution Products Functional Overview

Oracle Utilities meter solution products provide functionality for handling large volumes of meter and device data that enable increased accuracy, flexibility, and scalability. The following outlines the most crucial business processes that are enabled within the system:

- Defining meters, meter configurations, service points, and meter installations.
- Loading meter readings and interval data from a head-end system or other source.
- Automatic validation, editing, and estimating measurement data.
- Robust editing capabilities for readings and interval data
- Calculating and publishing bill determinants, and other data, from measurement data to be used in external, down-stream systems; such as, billing or pricing.

Oracle Utilities meter solution products store a lot of important data; including, meters, service points, customer contacts, and everything in between.

Measurement, VEE, and Usage Calculation for Billing



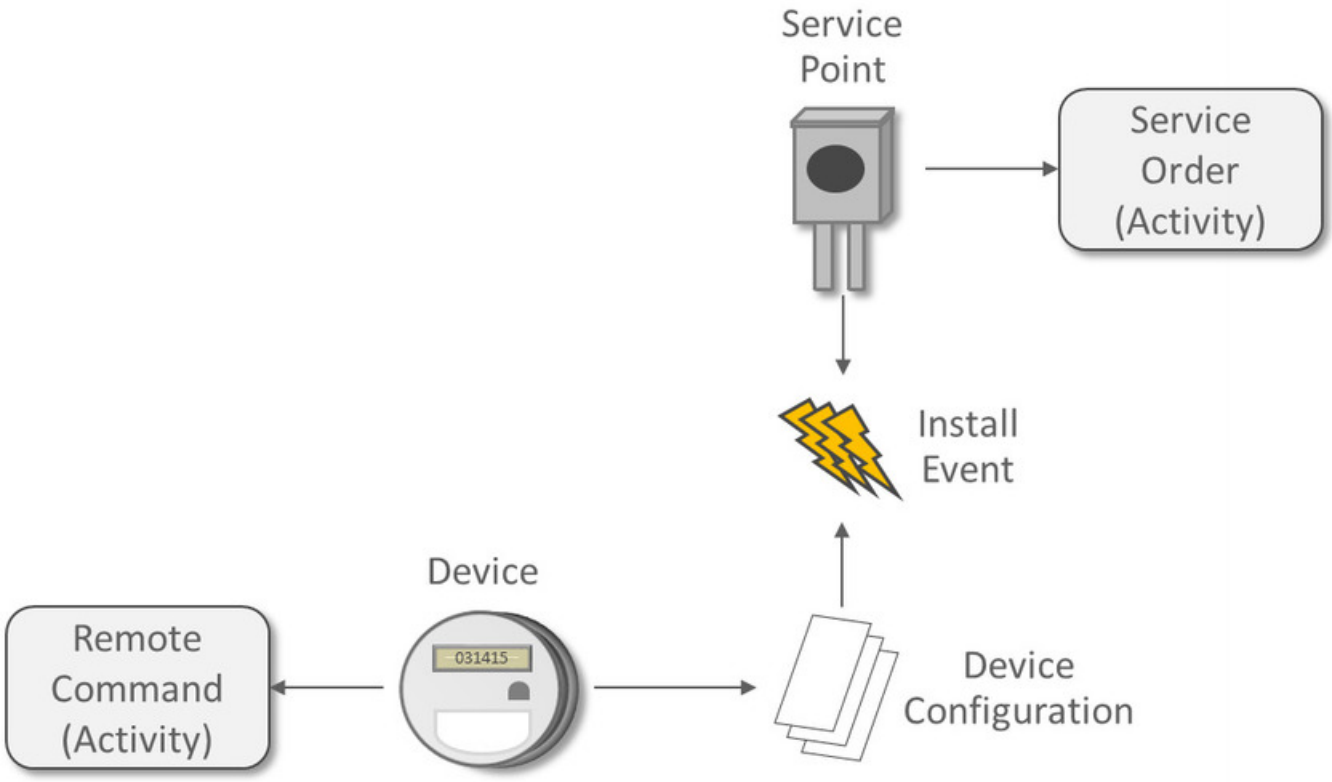
A Device represents a physical meter, communication module, or some other device in the field. A Service Point is the point where service is delivered to a customer and a device can be installed. The Install Event is a record of a specific device that is installed at a Service Point. A **Contact** is the customer that is associated with the Usage Subscription.

In order to properly track the way that a device is configured, the Device Configuration keeps a record of which types of data should be measured for the device. A Measuring Component represents a single channel of data for a device; for example, a device may have multiple measuring components: one that represents kWh interval data, another that represents kWh scalar readings, and a third that registers Voltage interval data.

Energy data received from meters is initially stored as Initial Measurement Data (IMD). Once the VEE process is executed, if the data passes then Measurements are created. If the data fails, however, then VEE Exceptions are created.

A Usage Subscription tracks a set of usage calculations that should be performed for a single or multiple Points. In order to perform calculation of usage (often referred to as bill determinants), a Usage Transaction is created through a request from the billing system. If any issues are encountered in the usage calculation process, or from a usage transaction validation, then Usage Transaction Exceptions are created.

Service Orders and Remote Commands



An Activity in Oracle Utilities Meter Data Management is a very flexible object that is used for various processes. Two of the key processes handled by Activities are:

- Remote Commands involve communicating with a Head End system to perform actions or retrieve data from a meter; for example, remote connects, remote disconnects, and on-demand reading.
- Service Orders are methods where work can be performed at a Service Point; for example, enabling service, disabling service, and meter exchanges.

Oracle Utilities meter solutions products include the following functional areas:

- **Device Management** is used by analysts and administrators to manage and define the devices used to record and capture meter data.
- **Device Installation** is used by analysts and administrators to manage device installation; including, defining markets and service providers, service points, contacts, and installation events.

- **Measurement Data** is a normalized way of storing data from a meter that involves some form of measurement (such as, kWh and CCF). Both interval data and scalar readings are held in this common storage location.
- **Validation, Editing, and Estimation (VEE)** is used by administrators to define validation, editing, and estimation rules to be applied to measurement data. VEE Exceptions may result from validation or estimation failures and should be worked by analysts through the To Do process.
- **360 Degree Search and View** is used by analysts and administrators to search and view data for devices, measuring components, service points, usage subscriptions, and contacts.
- **Consumption Sync** provides an automatic method to keep interval data and scalar readings in sync with one another.
- **Usage Calculation** is used by administrators to manage the calculation of usage data and to provide the results of those calculations (commonly referred to as bill determinants) to external systems and parties. Usage calculation groups and rules define calculation rules to be applied to measurement data. Usage Transaction Exceptions may result from usage calculation and should be worked by analysts through the To Do process.
- **Device Events** provide a view of specific events that have occurred on a meter. These are often unexpected and can indicate an issue with the meter.
- **Communication** helps track the instances when communication occurs with external systems. This is heavily used to track remote commands against Head End systems.
- **Aggregations** are used by analysts and administrators to search, view, and maintain aggregated measurements that represent a summarization of other measurements from a set of devices and/or measuring components.
- **Master Data Sync** defines the methods that data is automatically synchronized to Oracle Utilities Meter Data Management from external sources; such as, a CIS and/or Asset Management system.
- **Outage Storm Mode** is a way for Oracle Utilities Meter Data Management to detect widespread outages and suppress estimation for those meters until normal data communication resumes.
- **Dashboards** provide high level metrics for Oracle Utilities Meter Data Management operators to monitor operational trends as well as overall system health.
- **Service Issue Monitor** can be setup by administrators to monitor various conditions within Oracle Utilities Meter Data Management and automatically create a Service Investigative Order, if those conditions are met.
- **Service Order Management** provides a centralized program for managing the complex interactions required for Service Order processing. This area is especially valuable for managing service order processing that involves remote communication with a Head End for connects, disconnects, and on-demand readings.
- **Market Settlement** provides the core functionality for calculating and settling energy changes.
- **Information Lifecycle Management** is an automated method that administrators can configure to prepare data for archiving or purging after a defined period of time for the record type.

Architectural Overview

Oracle Utilities meter solution products are used to maintain information about meters and the service points at which they are installed. The applications provide means of recording measurements and events associated with meters in the field as well as the ability to compute usage for the recorded measurements.

Oracle Utilities meter solution products comprise the following functional areas:

- Device Management: the maintenance of physical meters in the field
- Device Installation: the maintenance of service points and the installation of meters in the field. This includes the means of registering outside systems to Oracle Utilities Meter Data Management for provider/consumer-specific processing of meter events and activities
- Device Communication: the maintenance of communications between Oracle Utilities Meter Data Management and head-end systems, including import of usage and events.

- Validation, Editing, and Estimation: the maintenance of measurement data and the engine used to validate and modify that data as it comes in
- Usage Management: the engine that calculates billable usage recorded on devices, applying factors and dividing the usage into configurable time of use periods
- Aggregation: summarization of measurement data for reporting and analysis purposes
- Data Synchronization: synchronizing data between Oracle Utilities Meter Data Management and other Oracle Utilities applications (including Oracle Utilities Operational Device Management and Oracle Utilities Customer Care and Billing).
- Oracle Utilities DataConnect: extraction of consumption and master data for use in external systems.
- Settlement: calculation and reconciliation of energy charges within a market

Oracle Utilities Meter solution products are built upon the Oracle Utilities Service and Measurement Data Foundation, a framework that provides shared functionality used by Oracle Utilities Meter Data Management, the Oracle Utilities Smart Grid Gateway adapters, and other Oracle Utilities products. Oracle Utilities Meter Data Management and the Oracle Utilities Service and Measurement Data Foundation are built atop the Oracle Utilities Application Framework.

Naming Conventions

Oracle Utilities Meter solution products use naming conventions to identify and distinguish entities that belong to different Oracle applications. These conventions can help you locate entities and understand their context.

Each base product uses a 2-character prefix for all its entities. For Oracle Utilities Meter solution products, these prefixes are as follows:

Product Name	Prefix
Oracle Utilities Application Framework	F1
Oracle Utilities Service and Measurement Data Foundation	D1
Oracle Utilities Meter Data Management	D2
Oracle Utilities Smart Grid Gateway for Landys+Gyr	D3
Oracle Utilities Smart Grid Gateway for Echelon	D4
Oracle Utilities Smart Grid Gateway for MV-90	D5
Oracle Utilities Smart Grid Gateway for Sensus RNI	D6
Oracle Utilities Smart Grid Gateway for Silver Springs Networks	D7
Oracle Utilities Smart Grid Gateway for Itron OpenWay	D8
Oracle Utilities Smart Grid Gateway for Adapter Development Kit	DG
Oracle Utilities Customer Self Service	WX
Oracle Utilities Meter Data Management Demo Data*	DM

*Note: this is not actually a product but rather a set of data that demonstrates Oracle Meter Data Management functionality.

Oracle recommends that you follow these naming conventions and develop your own set of conventions for the entities you create. If you create new entities, DO NOT use these prefixes; use the prefix "CM" (or some other unique prefix) to identify entities that have been customized.

High Level Functional Areas

A high level overview of the functional areas for Oracle Utilities Meter solution products has been defined in the Functional Overview section of the *Oracle Utilities Meter Solution Business User Guide*. This may be worthwhile to review to become better oriented to the areas of the Administrative User Guide before reading on.

Configuration Setup Sequence

This section provides the suggested order for the setup of Admin Data.

	Entity	Description
1	Feature Configuration	These are options that can be set to impact system processing. Typically these have wide ranging impact across several modules (e.g., multi-time zone support)
1	Country	Your organization's country.
1	Currency	Your organization's native currency.
1	Display Profile	Controls how dates, times, and numbers displayed.
1	Language	The language to use for this implementation.
1	Time Zone	Your organization's base time zone.
1	Work Calendar	The work calendar for your organization, which identifies your public holidays.
2	Installation Options	Control various global aspects of the system.
2	To Do Role	Used to associate users with To Do entries.
2	To Do Type	Used to define types of To Do Entries.
3	User	Defines a user's user groups, data access roles, portal preferences, default values, and To Do roles.
3	User Group	A group of users who have the same degree of security access.
3	Service Type	Defines specific types of service for which usage can be recorded and captured (electric, gas, steam, etc.).
4	Division	Delineates between different operating companies within a large conglomerate of utilities.
4	Unit of Measure	Quantities measured and recorded by the system (CCF, KWH, KW, etc.).
4	Service Quantity Identifier	Used to further distinguish between measured quantities that have identical UOM/TOU combinations.
4	Time of Use	Modifiers for a given unit of measure that indicate a period of time during which a quantity has been used (On- Peak, Off-Peak, etc.).
4	Factor	Centrally stored sets of values for use in validation rules, bill determinants calculations, and other processes.
4	Market	Defines jurisdictions or regulatory environments in which a Service Point participates.
4	Measurement Cycle	Defines the schedule for manual meter reading of devices at Service Points in that cycle
4	Measurement Cycle Schedule	Define the dates on which devices are scheduled to be read for a given measurement cycle.
4	Outbound Message	Defines messages sent to external systems.
4	Processing Timetable Type	Defines types of schedules that can be referenced by different processes and objects.

5	External System	Defines External Systems with which Oracle Utilities Meter Data Management should be able to communicate.
6	Head End Systems External Applications Market Participants	External entities that serve various roles relative to the application (head-end systems, billing systems, market participants, outage management systems, etc.).
6	Contact Type	Defines properties of a class of entities (businesses, persons).
7	Activity Type	Defines properties common to a specific type of activity.
7	Communication Type	Define properties common to a specific type of communication.
7	Service Task Type	Defines specific types of tasks performed by external users (self-service meter reads, self-service outage notifications, etc.)
7	Dynamic Option Type	Defines information common to dynamic options of a specific type.
7	Manufacturer	Individual companies that makes devices. Manufacturers also reference models.
7	Exception Type	Defines properties common to VEE Exceptions of a specific type.
8	VEE Group	Collections of VEE Rules that are applied to initial measurement data.
9	VEE Rule	Standard and custom VEE Rules that perform checking and/or manipulation of initial measurement data.
10	VEE Eligibility Criteria	Dictates whether a VEE Rule can execute based on a set of defined criteria.
11	Measuring Component Type	Defines the most important properties of a measuring component.
11	Measuring Component Comparison Type	Defines details by which measuring component data can be compared to determine the days that most closely resemble a specific day being evaluated.
12	Device Configuration Type	Defines properties of Device Configurations of a given type.
13	Device Type	Defines information about a class of devices.
14	Service Point Type	Defines specific types of points at which service is delivered.
14	Service Point Quantity Type	Defines types of quantities that can be stored for a service point.
14	Usage Subscription Quantity Type	Defines types of quantities that can be stored for a usage subscription.
15	TOU Group	Used to limit the set of Time Of Uses that are usable in a TOU schedule.
16	TOU Map Template	Schedules used for TOU map data generation.
17	TOU Map Type	Define important properties of TOU maps of a given type.
18	Usage Transaction Exception Type	Defines properties common to Usage Transaction Exceptions of a specific type.
18	Usage Calculation Group	Collections of usage calculation rules that are applied to measurement data to calculate bill determinants for Usage Subscriptions.
19	Usage Calculation Rule	Defines rules that perform calculations on measurement data to generate bill determinants and other values used by external systems.

20	Usage Calculation Rule Eligibility Criteria	Dictates whether a usage calculation rule can execute based on a set of defined criteria.
21	Usage Subscription Type	Defines collections of properties common to a set of Usage Subscriptions.
22	Data Source	Defines the source of data for dynamic aggregation, such as measurement data from usage subscriptions linked to a service point, badged or unbadged items, or measuring component sets
22	Dynamic Aggregation Measuring Component Type	Define the most important properties of a measuring component used with dynamic aggregation.
22	Measuring Component Set	Define the dimensions and criteria by which dynamic aggregation will be performed.
22	Aggregation Group	Define the ordering of a series of related aggregations based on a set of configured measuring component Sets.
23	Settlement Subscription Type	Defines collections of properties defining a class of settlement subscriptions.
23	Settlement Unit	Defines the “lowest common denominator set” of dynamic aggregation dimensions.
23	Settlement Calculation Group	Defines collections of settlement calculation rules that are used to perform settlement calculations, including calculating service point quantities, totaling consumption and usage for settlement units, application of losses, and allocation of unaccounted for energy (UFE).
23	Settlement Calculation Rule	Defines rules that perform settlement calculations, including calculating service point quantities, totaling consumption and usage for settlement units, application of losses, and allocation of unaccounted for energy (UFE).
23	Settlement Item Type	Defines specific types of end-customer settlement accounts.
23	Market Contract Type	Defines specific types of market contracts used in settlement processing.
23	Market Product Set	Defines the highest level grouping for a set of market products that will be processed together.
23	Market Product Type	Define types of products used in settlement processing.
23	Attribute Data Snapshot	Define types of attribute data snapshots used in settlement processing.
23	Measurement Data Snapshot	Define types of measurement data snapshots used in settlement processing.
24	Processing Methods	Control various behaviors for external applications, head end systems, and market participants within the system such as which message is sent, how an external value is translated, among others.
24	Master Configuration	Configuration that applies to series of modules that acts as a central point of configuration rather than embedding repetitive configuration throughout a set of algorithms.
24	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.

24	Analytics	Defines the extract parameters, the bucket configurations and configuration snapshots used for extracting data for Oracle Utilities Analytics

Chapter 3

Additional Resources

How to Get Support

Oracle is deeply committed to technical support services for its products. The Oracle Support online portal provides a reliable, easy-to-use method for obtaining technical support for Oracle Utilities Meter Data Management. To register as a new user for Oracle Support, go to https://support.oracle.com and follow the steps shown on the screen.

Oracle policies for standard technical support services are available at http://www.oracle.com/us/support/policies/index.html.

Knowledge Base Articles

Once you have access to Oracle Support, finding Knowledge Base articles is an easy process.

1. Log in to https://support.oracle.com.
2. Select the **Knowledge** tab at the top of the Oracle Support portal.
3. Under the Knowledge Base section, type "**Oracle Utilities Meter Data Management**" as the product.
4. Enter key search topics into the **Enter search terms** box.
5. Click **Search**.

The search results will display all knowledge articles stored on Oracle Support related to Oracle Utilities Meter Data Management for your key topics.

Important Articles

While there is a vast library of important Knowledge Base articles on Oracle Support, the list below highlights a few of the key knowledge articles that should be helpful if you're looking for additional detailed information about Oracle Utilities Meter Data Management:

Article Name	Knowledge Base Article Number
Information Center: Meter Data Management and Smart Grid Gateway Version 2	1907636.2
Related Measuring Component Consumption Sync White Paper	1672461.1
Overview and Guidelines for Managing Business Exceptions and Errors (this document was created by the CC&B team but pertains to MDM as well)	1628358.1

Support Hot Topic Emails

Oracle Support provides a way to receive daily or weekly updates on new information posted for Oracle Utilities Meter Data Management.

1. Log in to https://support.oracle.com.
2. Select the **Settings** tab at the top of the Oracle Support portal. (Note: this tab may be hidden for you and may require clicking a dropdown on the right side of the tabs)
3. Once on the Settings screen, choose the **Hot Topics E-mail** from the set of options on the left.
4. Under Delivery Options, choose **Send With Selected Options**.
5. Choose your delivery frequency: either Daily or Weekly.
6. Add content which interests you: there are a number of options available. To subscribe to updates for Oracle Utilities Meter Data Management, hit the **Add...** button under Selected Products. Enter "Oracle Utilities Meter Data Management" for the Product then choose the types of information for which you'd like to receive an email.
7. Make sure you click **Apply** at the bottom once you've made all of your selections.

Embedded Help

Oracle Utilities Meter Data Management, like all Oracle Utilities Application Framework applications, provides extensive internal documentation. For example, detailed descriptions of system objects are included in the objects' maintenance portals. The lifecycle of each business object is described on the Lifecycle tab and depicted in flow diagrams on the Summary tab. This information is extremely useful for implementers and system administrators.

Embedded help is provided for all non-obvious fields in most portals and zones. If a field has associated help text, a question mark icon appears next to the field when the zone is displayed.

Leveraging Demonstration Data

One of the best ways to understand Oracle Utilities Meter Data Management is through a thorough review of the demonstration data (also referred to as "demo data") provided by Oracle. The demo data for Oracle Utilities Meter Data Management provides the following:

- An overall view of how to implement the product for common needs
- Examples of the productized solutions to solve key utility scenarios
- Some examples of adding "customer modifications" on top of Oracle Utilities Meter Data Management

This demo data can be downloaded along with the Oracle Utilities Meter Data Management. It's recommended that this be installed into an isolated environment for reference purposes separate from other development or testing environments.

Once installed, the demo data scenarios can be found in two places within Oracle Utilities Meter Data Management:

- A zone named **Demo Scenarios** is shown in the right side Dashboard. This provides a quick method to link to the 360 Degree View for the specific demo meter.
- An extendable lookup named **Demo Scenarios Catalog** describes each of the scenarios in detail. This matches the scenarios displayed in the Demo Scenarios zone.

NOTE: Oracle recommends that you do not clone the demonstration environment as a basis for a new production environment. The demonstration environment typically includes transactional data that will be irrelevant to your production environment and can cause unexpected issues if it is not purged correctly. The recommended process is to start a new production environment from a new installation and migrate "clean" system data (such as business objects and algorithms) and administrative data (such as sample activity types or other administrative entities) from the demonstration and/or test or development environments as applicable. Instructions for using these tools are contained in the Bundling and Configuration Migration Assistant sections in the Oracle Utilities Application Framework *Administrative User Guide*.

Customer User Groups

A number of Customer User Groups have been established (one in the United States and others internationally) that are specific to sharing best practices and learnings about Oracle Utilities Meter Data Management and Oracle Utilities Smart Grid Gateway (SGG). If you're interested in participating in a Customer User Group, contact the appropriate Product Manager.

Chapter 4

Best Practices

Performance Recommendations

Initial Measurement Loading Recommendations

- In general, keep the number of devices to 1,000 per file for optimum usage and event processing through OSB. A lower number of devices, per file, will take more time for processing. A higher number of devices, per file, leads to high growth in garbage collection leading to waits and results in lower throughput. Please note, the optimum number of transactions per file may vary by head-end system.
- Initial measurement payloads should have very selective criterion to get to the exact measuring component (MC). In an ideal case, it should be the MC identifier along with device serial number. If this is not provided, then the UOM/TOU/SQI configured on the MC type is used to retrieve the exact business object (BO) from the Service Provider which might have the same values for multiple channels.
- Populating the raw data section of the IMD record will reduce overall throughput

VEE Recommendations

- For the High-Low VEE Rule, the historic pre window should be set as low as possible based on business needs. This rule has an intensive hit on performance of the Initial Measurement loading process.
- Each VEE Rule included in VEE processing can have an impact on overall performance. The table below summaries the relative costs in terms of performance for several commonly used VEE Rules. Using this table when designing your VEE groups and rules can help identify potential performance challenges.
- For example, using "Minimal Impact" rules will likely have little to no significant impact on performance, while using several "Moderate Impact" rules could result in performance challenges, and using the High Low Check rule is very likely to introduce noticeable impacts to VEE processing performance.

Relative Cost	VEE Rules
Minimal Impact (0% - 5%)	Interval Interpolation Interval Replacement Rule Interval Size Validation Interval Spike Check Negative Consumption Check Scalar Replacement Rule UOM Check
Moderate Impact (5% - 20%)	Final Measurement Validation Interval Adjustment From Scalar Interval Averaging Multiplier Check Scalar Calculation From Interval Scalar Proration Sum Check
Significant Impact (20%)	High/Low Check

- Consumption sync processes should only be run on devices where the difference in consumption is outside the defined tolerance as dictated by the sum check tolerance. In a production environment, running consumption sync for two percent of devices or less is advisable.
- Using Scripting for simple VEE Rules is approximately 10% more expensive than a similar Java rule.
- Run performance testing on all configuration changes performed that impact portions of Oracle Utilities Meter Data Management with high transactional processing.
- Any VEE Rules, whether provided as part of base product or custom developed, should be configured to query no more than 30 cumulative days of historical data (i.e. High/Low) to help optimize the data loading process
- A good way to troubleshoot a VEE Rule is to use the Trace section of the IMD Log. When an IMD is in Pending status, the **Trace On** button can be used to trace all VEE Rules fired during IMD processing. Once processing is complete, view the Log tab of the IMD and review the results in the **IMD Trace Log** section. For any custom developed VEE Rules, the average run time should be no longer than the average run time of other base product VEE Rules. Also, the impact of different configuration can be checked through this method as well.

Usage Transaction Recommendations

- Saving derived interval data and sending the data to an external system via the "Save Interval Vector" and "Extract Interval Data" options on the "Vector and Service Quantity Math" and "Get Interval Data" usage calculation rules can have a noticeable impact on usage transaction processing. Usage transaction processing can be improved by up to 45% if extraction of interval data is eliminated. Use of these options should be limited to situations in which they are absolutely required.
- For Outbound Messages, the schema extraction for usage transactions should be converted into a RAW string operation to allow for more efficient Pre-Processing logic.

User Interface Recommendations

- For better performance, user interface zones should be initially collapsed when not required for 90% or more of business processes. The initial state of zones (collapsed or not) can be controlled via the "Portal Preferences" tab on the **User** portal.

- The number of records returned to the user interface for a zone should be limited to 50 rows when building custom zones against large transactional tables.
- UI Map schemas should be specific to the displayed data. This will make sure that application will not visit all the elements that are not required for display.
- A good way to troubleshoot a screen in the user interface is to go to the **Preferences** in the top right hand corner of the application and choose the **Portal Preferences** tab. Choose the appropriate portal and set all zones to "Initially Collapsed". Next, navigate back to the screen that has performance issues, expand the zones one by one, and measure the execution time of each zone. This should be an accurate step-by-step representation of the full screen execution. This testing is especially important for **custom** user interfaces.
- For optimal user interface performance, Oracle Utilities Meter Data Management users should ensure their computer set to **high performance mode** so all CPU resources are used. Also, check if allocating additional memory through your browser's default settings helps improve performance.

SQL Recommendations

- Avoid making unbounded SQLs statements with no boundary condition on date columns.
- Reduce CLOB searches and use physical columns wherever possible.
- Implement caching to pre-fetch data instead of issuing multiple SQLs.

Java Recommendations

1. For areas where the transactional volume is high (such as Initial Measurement), use Hibernate SQL for non-CLOB fields whenever possible as opposed to reading entire Business Object. If a CLOB field must be retrieved, then either Entity or Lite BO should be used. Reading the full Business Object should be used as the last resort for high volume areas. This process should be considered for updating data as well when it provides the same functionality as updating the Business Object. These methods should only be considered if BO level validations, pre-processing, and post-processing aren't required.
2. For a simple SQL select statement needed in Java (not many joins and no complex logic), using Hibernate SQL provides a benefit over using a Business Service and a Zone since the entities are cached for Hibernate SQL.
3. "Lite" Business Objects are provided as a way to access the main fields for a BO without pulling back all of the information. Retrieving less information will speed up the process for reading the BO.

Referencing Master Data by Identifiers

Understanding Referencing Master Data by Identifiers

There are many places within admin configuration where direct references to master data can be made. Since master data relies on system generated keys this configuration often breaks once migrated to a new environment since the master data referenced does not exist in the target environment. To alleviate this issue, in each of these instances, a user defined identifier can be used instead of the system generated keys.

This has an added benefit for installations that support multiple time zones by enabling the identification of the master data to not only search by the user defined identifier but also the time zone of the instance data. This is advantageous because it allows one set of admin configuration to satisfy master data in multiple time zones since things like TOU Maps and profile measuring components will be identified at run time using the identifier and time zone.

The following types of data can be referenced by an identifier on admin configuration:

- **Device**: can be referenced by a device identifier type. An example of this can be seen on the Specification Lookup (D1-SpecificationLookup) extendable lookup.
- **Measuring Component**: can be referenced by a measuring component identifier. When multiple time zone support is enabled the measuring component's time zone will also be evaluated. There are two major patterns for this:
- *Profile Factors*: The profile factor itself will reference the measuring component by identifier and then the admin configuration will point to the profile factor. An example of this is the profile factor list on interval measuring component types.
- *Direct Measuring Component references*: An example of this is the Final Values Overlay Profile (D1-FinalValuesOverlayProfile) extendable lookup.
- **TOU Map**: can be referenced by the TOU map template code. The appropriate TOU map is then found by searching for the appropriate TOU map that references a TOU map type for the supplied TOU map template. When multiple time zone support is enabled the TOU map type time zone will also be evaluated. An example of this is the default TOU map that can be configured in the display profile of a measuring component type.

During execution the identifier supplied will be used to find the appropriate master data entry. The following validation will occur:

When multiple time zone support is off: the search must find one and exactly one master data entry for the identifier type and value.

When multiple time zone support is on: the search must find one and exactly one master data entry for the identifier type, identifier value, and time zone. Note: some installations may want to use master data across time zones, this can still be achieved by supplying one and only one master data entry for the identifier type and value:

When the search fails to find the master data for the identifier type, identifier value, and time zone and there is only one entry for the identifier type and value that master data will be used (thus allowing one master data entry to be used across time zones)

When the search fails to find the master data for the identifier type, identifier value, and time zone and there are multiple entries for the identifier type and value an error will be encountered.

Recommendations for Creating a Production Environment

Oracle recommends that you do not clone the demonstration environment as a basis for a new production environment. The demonstration environment typically includes transactional data that will be irrelevant to your production environment and can cause unexpected issues if it is not purged correctly. The recommended process is to start a new production environment from a new installation and migrate "clean" system data (such as business objects and algorithms) and administrative data (such as sample activity types or other administrative entities) from the demonstration and/or test or development environments as applicable.

Your implementation can use bundling and/or the Configuration Migration Assistant to move system and administrative data. Instructions for using these tools are contained in the Bundling and Configuration Migration Assistant sections in the Oracle Utilities Application Framework *Administrative User Guide*.

Contact Oracle customer support if further assistance is required.

Chapter 5

System-Wide Options

Installation Options - Framework

Configuring Installation Options - Framework

Installation options define the individual applications installed on your system and identify algorithms used to implement core system functions. These options also define global parameters such as the administrative menu style (alphabetical or functional), the country, language, currency code, as well as the base time zone to use for this implementation.

Installation options are stored in the installation record for your system. Use the Installation Options - Framework portal to configure these options. This portal is part of the OUAF and is described in detail in the Framework documentation.

Base Time Zone

The time zone setting of the Installation Options - Framework determines the time zone for all date/times stored within the system. Each date/time, based on the configuration of that field, is stored in either standard or legal time within this base time zone.

Refer to the Glossary of Terms in the *Oracle Utilities Meter Solution Business User Guide* for definitions Standard Time and Legal Time

Note: The installation record does not dictate the server time zone, but rather must match it.

Installation Algorithms

Installation algorithms implement global system functions and can be customized for each implementation. The base package supports the following installation options for Meter Data Management-related system events:

- Geocoding Service: Responsible for geocoding an address (converting an address to a geocode latitude/longitude pair).
- Global Context: Sets global contexts (displayed in the Global Context dashboard zone) based on the value of existing global contexts. For example, if the Service Point is specified, this algorithm sets the Device by finding the most recently

installed Device on the service point. It then sets the Measuring Component by finding the most effective Device Configuration and retrieving any measuring component linked to it. It then sets the Usage Subscription by finding the most recent active usage subscription linked to the service point. The contact is set by finding the main contact for the usage subscription.
Additional detail on how global context is populated can be found in the detailed description of the D1-GBCTX-DF Algorithm Type.

- To Do Pre-creation: Associates a To Do entry via characteristic to the related the related Device, Measuring Component, Service Point, Contact, Usage Subscription, Activity Type and Activity based on the drill keys of the To Do entry.

  NOTE:
  Additional detail on how the To Dos are associated to related data can be found in the detailed description of the D1-TDPRCRE algorithm type.

  See Installation Options in the Oracle Utilities Application Framework *Administrative User Guide* for related information on the installation portal.

Feature Configurations

Configuring Feature Configurations

Some of the features in Oracle Utilities Application Framework based applications are configured by populating options on a "feature configuration". For example, if your implementation uses Oracle Utilities Customer Care and Billing's batch scheduler, you must populate a variety of options on the batch scheduler's feature configuration.

NOTE: Refer to Defining Feature Configurations in the Oracle Utilities Application Framework *Administrative User Guide* for additional information.

Oracle Utilities Meter Data Management uses the following types of feature configurations (please note that more information can be found on each of these options by viewing their detailed description in the Feature Configuration portal):

Measurement Data Options

Measurement Data Options are used to define behavior related to periodic estimation of initial measurement data, including:

- **No of Hours in Past to Retrieve Last Usable Measurement**: This option is leveraged by scalar periodic estimation to restrict how far into the past it will search for existing measurements when the last contiguous measurement is being initialized on the measuring component. This is to ensure that the first time scalar period estimation is executed on a large number of measuring components that have not been initialized for periodic estimation the system does not execute a large number of unbounded queries into the past which would result in poor performance.

Business Intelligence Configuration

Business intelligence configuration is used to define external data source indicators used when Oracle Utilities Meter Data Management is integrated with Oracle Utilities Business Intelligence. External data source indicators allow business intelligence extracts to properly link the external identifiers to the source external system. The Value of the Data Source Indicator option should match the Environment ID on the Installation Option of the external system.

General System Configuration

This feature configuration is owned by Oracle Utilities Application Framework but there are several important option types that have been created specific to Oracle Utilities Meter Data Management:.

- **Multi Time Zone Support**: By default the system assumes an installation operates within a single time zone. In order to enable multiple time zone functionality this option must be defined and set to "D1YS".
- **CCB Link URL**: this option is used to provide the destination URL for hyperlinking into Oracle Utilities Customer Care and Billing.
- **Trace On Flag**: can be set to automatically trace any initial measurement being processed. Note: if the initial measurement has had trace explicitly turned off then no tracing will occur. This is useful for tracing initial measurements created through automated processes where you are unable to set the trace flag directly.
- **System Override Date**: this option type is provided by OUAF but it is highly useful in testing prior to production. When set it will override the system date/time for all users. The format must be entered as: YYYY-MM-DD. For example January 1st 2010 would be 2010-01-01.

NOTE:
Refer to Configuring Multi-Time Zone Support for additional information on setting up multi-time zone support.

Time Zones

Configuring Time Zones

To support businesses spanning across multiple time zones, the system stores all date and time information in a single common time zone known as the base time zone or the server time zone. Furthermore, date and time information is stored in either standard time (i.e. independent of any Daylight Savings Time adjustments, if applicable, in that time zone) or legal time (i.e. shifted according to Daylight Savings Time).

The system also allows data to be entered and displayed in a different time zone in legal time (i.e. adjusted for Daylight Savings Time, managing the conversion back and forth between the data entry time zone and the storage time zone).

Entities associated with a physical location such as measurements (initial and final), measuring components, device configurations, devices, installation events, service points, and usage subscriptions are entered and displayed in the specific time zone where they occur, the entity time zone. The rest of the application uses the base time zone to display date and time information.

When configuring time zones the following fields are of high importance:

- **Time Zone Name**: identifies the Olson time zone and as such defines the appropriate offset from Greenwhich Mean Time as well as the schedule for shifting into and out of Daylight Savings Time.
- **Default Time Zone Label**: will be appended to date times that do not fall within Daylight Savings Time
- **Shifted Time Zone Label**: will be appended to date times that do fall within Daylight Savings Time

NOTE: The server time zone must be correctly specified on the installation options record for the system to work properly.

For additional information see Defining Time Zones in the *Oracle Utilities Application Framework Administrative User Guide*.

Configuring Multi-time Zone Support

As a default the system assumes operations are in a single time zone. This has a few high level impacts:

- All master data must have a time zone that is equal to the base time zone. As such all master data will have the same time zone and that time zone will be the base time zone
- Certain logical complexities are avoided during high volume processing given the knowledge that all master data time zones are equal to the base time zone.

If your implementation resides within multiple time zones then the Multi Time Zone Support feature configuration must be enabled. Doing so will allow master data to be defined with any time zone configured in the system. It will also enable logic to convert time zones between the master data time zones and the base time zones.

You can access the feature configuration portal from the **Admin > General > Feature Configuration**.

From the list of results returned select the feature name for the feature type General System Configuration. For the option type Multi Time Zone Support supply "D1YS" as the value to turn multiple time zones on.

Refer to Multiple Time Zone Support for information about how functional processing is impacted by multiple time zone support.

Chapter 6

Defining General Options

Units of Measure

Understanding Units of Measure

Units of Measure (UOM) identify quantities measured and recorded, such as KWH, KW, cubic feet, degrees Celsius, etc. UOMs are based on a specific service type.

Important Unit of Measure System Events

Unit of Measure supports the following business object algorithm system events:

- **Axis Conversion**: this event receives a list of measurements along with a source UOM and interval size and a target UOM and interval size. It should then perform the necessary actions to convert the source UOM and interval size into the target UOM and interval size. Refer to the algorithm Axis Conversion algorithm (D1-AXIS-CONV) as an example.

Configuring Units of Measure

This portal is used to display and maintain a Unit of Measure.

Refer to Understanding Units of Measure for more information.

You can access the portal by selecting **Admin**, then **General**, then **Unit Of Measure**.

The following zones may appear as part of the portal's **Main** tab page...

- **Unit of Measure List:** This zone lists all units of measure. Broadcast a record to display the details of the selected record.
- **Unit of Measure:** This zone provides information about the selected unit of measure.

Service Quantity Identifiers

Understanding Service Quantity Identifiers

Service Quantity Identifiers (SQI) are used to further distinguish between measured quantities that have identical UOM/TOU combinations, including situations in which the distinguishing identifier of a UOM is not accurately described as a TOU. Some examples include: Generated, Consumed, etc.

SQIs can also be used as a stand-alone representation of a service quantity that is not measured (one that is not properly described as a UOM) within a usage service quantity collection (such as a billing determinant).

Configuring Service Quantity Identifiers

This portal is used to display and maintain a Service Quantity Identifier.

Refer to Understanding Service Quantity Identifiers for more information.

You can access the portal by selecting **Admin**, then **General**, then S**ervice Quantity Identifier**.

The following zones may appear as part of the portal's **Main** tab page:

- **Service Quantity Identifier List:** This zone lists all Service Quantity Identifiers. Broadcast a record to display the details of the selected record.
- **Service Quantity Identifier:** This zone provides information about the selected Service Quantity Identifier.

Time of Use

Understanding Time of Use

Time of Use (TOU) periods are modifiers for a given unit of measure that indicate a period of time during which a quantity has been used, such as On-Peak (meaning during a time when the greatest quantity of some consumable is being used), Off-Peak (meaning during a time when the least amount of some consumable is being used), etc.

Configuring Time of Use

This portal is used to display and maintain a Time of Use.

Refer to Understanding Time of Use for more information.

You can access the portal by selecting **Admin**, then **General**, then **Time Of Use**.

The following zones may appear as part of the portal's **Main** tab page:

- **Time Of Use List:** This zone lists all Time Of Use records. Broadcast a record to display the details of the selected record.
- **Time Of Use:** This zone provides information about the selected Time Of Use.

Service Types

Understanding Service Types

Service Types define specific types of service for which usage can be recorded and captured, such as electric, gas, steam, etc.

Configuring Service Types

This portal is used to display and maintain a Service Type.

Refer to Understanding Service Types for more information.

You can access the portal by selecting **Admin**, then **General**, then **Service Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Service Type List:** This zone lists all Service Type records. Broadcast a record to display the details of the selected record.
- **Service Type:** This zone provides information about the selected Service Type.

Divisions

Understanding Divisions

Divisions are used to delineate between different operating companies within a large conglomerate of utilities.

Divisions can have one or more characteristics defined.

Divisions can be defined on several other objects and entities in the system, including:

- Service Point
- Usage Subscription/Usage Subscription Type
- Usage Calculation Group
- Settlement Calculation Group
- Aggregation Group
- Data Source
- Market Product Type
- Market Product Set
- Measurement Data Snapshot Type
- Attribute Data Snapshot Type

Configuring Divisions

You use the **Divisions** portal to display and maintain divisions.

Refer to Understanding Divisions for more information.

You can access the portal by selecting **Admin**, then **General**, then **Division**.

The following zones may appear as part of the portal's **Main** tab page:

- **Division List**: This zone lists all division records. Broadcast a record to display the details of the selected record.
- **Division**: This zone provides information about the selected division.
- **Characteristics**: This zone displays a list of characteristics defined for the selected division.

Factors

Understanding Factors

Factor are a centrally stored set of values for use in validation rules, bill determinants calculations, and other processes.

A factor can have different values depending upon some definable attribute of a system object, such as customer size associated with a service point. Examples of factors can include minimum and maximum thresholds, loss factors, etc. Classes of factors are defined that can have numeric values (as in the above examples), or values pointing to profile measuring components, or VEE groups.

A factor's values are effective-dated values - either a number, a profile measuring component, a temperature profile measuring component, a VEE group, or some custom-defined value - assigned to a factor and associated to the value of some attribute of a system object. For example, consider a service point that can be classified as residential, commercial, or industrial. The tolerance percentage by which a customer's consumption can exceed last month's consumption can be based on the service point category.

For this example, factor values for a single factor called "tolerance percentage" could be:

- Residential - 20%
- Commercial - 10%
- Industrial - 5%.

Configuring Factors

This portal is used to display and maintain Factors and Factor Values.

Refer to Understanding Factors for more information.

Prerequisites: You must define factor characteristic source algorithms, factor characteristic types, and factor characteristic values before you can create a factor. Refer to the Oracle Utilities Application Framework online help for more information about algorithms, characteristic types, and characteristic values.

You can access the portal by selecting **Admin**, then **General**, then **Factor**.

The following zones may appear as part of the portal's **Main** tab page:

- **Factor List:** This zone lists all factor records. Broadcast a record to display the details of the selected record.

- **Factor:** This zone provides information about the selected factor.
- **Factor Char Value and Factor Value List:** This zone lists the characteristic values associated to the factor characteristic. For each characteristic value it will display the related factor values. From this zone you can click the **Factor Value** link to see the factor value in more detail on the **Factor Value** portal.

Markets

Understanding Markets

Markets define the jurisdictions or regulatory environments in which a service point participates.

Markets also define market relationships for valid service providers and their roles within a market (distributor, etc.). While each service point specifies only one market, a utility may serve more than one market, and different service points throughout the utility's service territory can be linked to different markets.

For each service provider defined for a market, you can also specify a fallback service provider.

Service Providers in Deregulated Markets

Some utilities operate in deregulated markets. In implementations in deregulated markets, the system can send information to and receive information from a variety of market entities. These entities are defined as service providers.

For example, a service point's distribution company and/or energy supply company may subscribe to its consumption, or a service point's meter service provider may send requests to ping the meter that's installed at the service point to verify connectivity between the meter and its head-end system.

Different Relationship Types in Different Markets

Each market can define different relationship types between its service providers. A single instance of Oracle Utilities Meter Data Management or Oracle Utilities Smart Grid Gateway may have service points in different markets where each market has different relationship types and service providers. For example:

- In a regulated market the distribution company is the de facto energy supplier and meter service provider.
- Another market might have two relationship types and a single service provider for each relationship:
  1. There is a single energy supply company for the entire market
  2. There is a single meter service provider for the entire market

Yet a another market might have two relationship types (energy supply and meter service). In this market, there might be multiple service providers for each relationship type. Each service point can choose any of the relationship type's service providers. If a service point does not declare a specific service provider for a given relationship type, the relationship type's "fallback" service provider is assumed.

Configuring Markets

This portal is used to display and maintain a Market.

Refer to Understanding Markets for more information.

You can access the portal by selecting **Admin**, then **Market**, then **Market**.

The following zones may appear as part of the portal's **Main** tab page:

- **Market List:** This zone lists all Market records. Broadcast a record to display the details of the selected record.

- **Market:** This zone provides information about the selected Market.

Market Participants

Understanding Market Participants

Market Participants are participants in a deregulated environment. Relationships between market participants are defined in a particular market record. Refer to Understanding Markets for more information.

Each market participant can be associated to an external system which is used to define the messages that can be sent to that market participant and how each message is sent.

Configuring Market Participants

This portal is used to display and maintain market participants.

Refer to Understanding Market Participants for more information.

You can access the portal by selecting **Admin**, then **Market**, then **Market Participant**.

The following zones may appear as part of the portal's **Main** tab page:

- **Market Participants List:** This zone lists all market participant records. Broadcast a record to display the details of the selected record.
- **Market Participant:** This zone provides information about the selected market participant.
- **Market Participant Characteristics**: This zone lists characteristics defined for the market participant.
- **Market Contracts**: This zone lists market contracts where the market participant is either the Buyer or Seller.
- **Processing Method List:** This zone provides the list of processing methods defined for the market participant.
- **Translation Method List:** This zone provides the list of translation methods defined for the market participant.
- **Inbound BOs Send By Service Provider:** This zone lists inbound business objects that are sent by this market participant. The identification is driven by the business object having a Business Object Option of type "Sent By Service Provider" that references the current market participant.

Understanding Processing Methods

Head end systems, external applications, and market participants can have one or more associated processing methods that define the format or means by which it receives or sends data from or to the application, such as bill determinants, interval data, or meter events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and usage transactions. Processing methods can also be used to define the information an external system wishes to subscribe to receive from our application.

At the lowest level processing methods are used to identify an outbound message type, business object, batch control, or message category and number.

Each processing method is comprised of a business object that defines what is being mapped as well as how it should be mapped.

Important Processing Method System Events

The actual logic to determine the appropriate output for a given head end system, external application, or market participant and processing role for a processing method is executed by the following system event:

- **Determine Processing Method(s)**: is a business object algorithm system event that takes in a head end system, external application, or market participant, a processing role, and a list of relevant input data aka related object (e.g. a measuring component, device, etc). For the head end system or external application and the input data it will analyze the selection criteria to determine the appropriate output.

How Processing Methods Work

Processing methods perform two basic tasks:

- They define the criteria for selecting the appropriate output. This can be as simple as providing a single object in return without qualification or in more involved situations it could support determining the appropriate return object based on characteristics of the data being processed. This is accomplished through the data structure defined on the processing method business object.
- They evaluate the criteria for selecting the appropriate output given a specific head end system, external application, or market participant. This is accomplished through the Determine Processing Method system event on the processing method business object.

Each processing method business object can be associated to one or more processing roles. This is done by adding the Applicable Processing Role business object option. It is these processing roles that actually create the association between a head end system, external application, or market participant, the processing method, and the functional event that is being executed. For example, when initial measurement data is processed through the IMD Seeder the processing role Initial Measurement Creation is used to identify the processing method "How to Create MC Related Information" which maps a particular measuring component type to the appropriate initial measurement business object to be used for processing.

When system logic requires the results of a processing method the service Determine Service Providers and Methods is used. This service is available to call via Java or from within scripting through the business service of the same name. It can be called in one of two ways:

With a head end system, external application, or market participant: identifies the appropriate output for a single head end system, external application, or market participant being processed for the input processing role and related objects

Without a head end system, external application, or market participant: used to identify subscribing sysems. This will provide a list of any head end system, external application, or market participant that has the input processing role and an appropriate output given the related objects.

Processing Methods Available

There are the following processing methods provided by the base package:

Name	Details	Business Object
How to Create OB COMM/Send OB Message	Identifies Message Number/Category, Business Object, Outbound Message Type and allows for an override by a device type.	D1-HowToCreateActivityOBComm
How to Create MC Related Information	Identifies a business object and allows for an override by measuring component type.	D1-HowToCreateMCInformation
How to Process Device Event Related Info	Identifies a business object, outbound message type, and batch control by device event category allowing for an override by device event type.	D1-HowToProcDvcEvtsInformation
How to Process Device Related Information	Identifies a business object and allows for an override by device type.	D1-HowToProcessDeviceInfo

How to Send Activity Related Information	Identifies a batch control and business object by activity type and allows for an override by device type.	D1-HowToSendActInformation
How to Send Activity Related O\B Messages	Identifies an outbound message type, message category, and message number and allows for override by activity type.	D1-HowToSendActivityResponse
How To Create US Related Information	Identifies a business object and allows for override by usage subscription type.	D2-HowToCreateUSInformation
How To Send US Related Information	Identifies a batch control, business object, and outbound message type and allows for override by usage subscription type.	D2-HowToSendUSInformation
How to Process Service Point Related Info	Identifies a business object and allows for override by service point type.	D1-HowToProcSPRelatedInfo
How to Send Field Activity Related Info	Identifies a outbound message type and allows for an override by field task type.	D1-HowToSendFARelatedInfo
How to Send Field Activity Remark Info	Identifies an outbound message type for an activity remark type.	D1-HowToSendActivityRemarkInfo
How to Translate External Value	Identifies a business object and allows for override by identifier.	D1-HowToTranslateExternalValue
How to Request Customer Notification	Identifies a list of outbound message types.	D1-HowToRequestCustomerNotific
How to Process Business Flag Related Info	Identifies an outbound message type and allows for an override by business flag type.	D1-HowToProcessBusinessFlagInf

Processing Timetable Types

Understanding Processing Timetable Types

Processing timetable types define types of schedules that can be referenced by different processes and objects in Oracle Utilities applications such as Oracle Utilities Meter Data Management and Oracle Utilities Market Settlements Management. For example, if a process requires requesting data from an external system on a daily basis, a daily processing timetable type could be used to define the details of this schedule.

Processing timetable types are used by standalone data request measuring components to control the schedule by which requests for data are processed (for example, in a settlement process in which zonal load is requested on a daily basis). They are also used in the attribute data snapshot creation process.

The base package includes two processing timetable type business objects used with specific scheduling patterns:

- **Daily Processing Timetable**: Used for daily schedules
- **Defined Dates Processing Schedule**: Used for schedules that should execute on specific dates.

Parameters used to define processing timetable types include:

- **Execution Frequency Class**: Designates the type of processing timetable, Daily or Defined Dates (based on the business object)
- **Scheduling Information**: Details about how the schedule is executed, including:

- **Period Cut-Off Time**: The time of day that represents the end of a period. In many instances this will be 12:00 AM. However, for something like a "Gas Day" this might be set to 09:00 AM.
- **Earliest Execution Time**: Used to prevent records from being processed until after this time each day. This time relates to the processing time.
- **Lag / Lead**: A flag that indicates whether periods in the past (Lag) or future (Lead) should be processed (applies to Daily schedules only)
- **Lag Days / Lead Days**: The number of days in the past (Lag) or future (Lead) that should be processed (applies to Daily schedules only)
- **Restrict Processing by Work Calendar**: Used to restrict processing to only the work days configured in the Work Calendar and will exclude any holidays added to the calendar (applies to Daily schedules only)

- **Workflow Information**: Details about how the processing timetable aligns with process workflows, including warning time and target due time
- **Processing Dates**: A list of specific dates on which the process should be executed (used with Defined Dates Processing Schedule only)

Configuring Processing Timetable Types

Refer to Understanding Processing Timetable Types for more information.

You can access the portal by selecting **Admin**, then **General**, then **Processing Timetable Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Processing Timetable Type List**: This zone lists all processing timetable type records. Broadcast a record to display the details of the selected record.
- **Processing Timetable Type**: This zone displays details for the selected processing timetable type.

Chapter 7

Defining Asset Options

NOTE: The asset management functionality described in this section is available only to cloud implementations using Oracle Utilities Customer Cloud Service or Oracle Utilities Meter Solution Cloud Service.

Asset Activity Types

Understanding Asset Activity Types

Activity types define the main attributes broad range of activities, such as maintenance, repairs, inspections, field work etc. The base package includes an asset activity type business object (W1-SOMActivityType) that should be used when created asset activity types.

Activity types are defined by:

- **Activity Business Object**: The business object used for asset activities of this type
- **Activity Schedule Information**: Default scheduling information for asset activities of this type, including Activation Lead Days, Work Lead Days, and Activity Disposition.
- **Attachments**: One or more attachments associated with asset activities of this type, such as procedures or checklists.
- **Exception Handling**: Details regarding exception handling for asset activities of this type, including default To Do Type, To Do Role, and retry options.
- **Completion Event Types**: One or more completion event types associated with asset activities of this type. See Understanding Completion Event Types for more information.
- **Object To Create**: The service history type to be created by asset activities of this type. Service history records are created by Create Activity Type's Service History completion events.

Refer to Defining Asset Activity Types for more information about setting up asset types.

NOTE: Asset activity types are created by the system when field task types are created in Service Order Management.

Defining Asset Activity Types

You use the **Activity Type** portal to display and maintain asset activity types.

Refer to Understanding Asset Activity Types for more information.

You can access the portal by selecting **Admin**, then **Asset Management**, then **Asset Activity Type**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Activity Type**: This zone displays basic information about the activity type.
- **Attachments**: This zone attachments associated with the activity type.

Asset Completion Event Types

Understanding Asset Completion Event Types

Asset completion event types are used to define various processes or events to be executed when completing an asset activity such as asset replication, service history creation, attachment creation, and activity creation.

Asset completion event types are referenced on activity types to indicate which completion events should be included as part of the lifecycle of activities of that type. For example, this affects when service history is created for an activity, and the disposition of the assets on an activity as the activity is processed. Asset completion events can be configured as optional or required for activities of different types.

Asset completion event types are defined by:

- **Completion Event Business Object**: The business object used for asset completion events of this type
- **Identifiers**: One or more identifiers that can be used to characterize asset completion events of this type
- **Inbound Communication**: Information used to create asset completion events of this type using information defined within an inbound communication business object, or within its raw message
- **UI Payload Create Data Mover**: XPath statements executed when creating asset completion events of this type online. These statements are used to populate elements within the payload from the completion event business object.

Refer to Defining Completion Event Types for more information about setting up completion event types.

Defining Asset Completion Event Types

You use the **Completion Event Type** portal to display and maintain asset completion event types.

Refer to Understanding Asset Completion Event Types for more information.

You can access the portal by selecting **Admin**, then **Asset Management**, then **Asset Completion Event Type**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Completion Event Type** : This zone provides information about the selected completion event type, including as inbound communications and attachments.
- **Unlink Activity Types**: This zone enables the removal of an activity type from the completion event type. To remove an activity type select the activity type(s) to remove and click **Remove**. The selected activity types will no longer appear in the **Unlink Activity Types** zone.
- **Link Activity Types**: This zone enables adding an activity type to the completion event type. To link an activity type:
  1. Enter search criteria n the **Activity Type Description** field and click **Search**. The zone will display a list of activity types matching your search criteria.
  2. Select the activity type(s) to add and click **Link**. The selected activity types will now appear in the **Unlink Activity Types** zone.

Asset Inbound Communication Types

Understanding Asset Inbound Communication Types

Asset inbound communication types define the information required for the system to receive messages from external systems. These typically hold value mappings of the inbound communication elements to an xpath element in the completion message.

Asset inbound communication types are defined by:

- **Inbound Communication Business Object**: The business object used for asset inbound communications of this type
- **Exception Handling**: Details regarding exception handling for asset inbound communications of this type, including default To Do Type, To Do Role, and retry options.
- **Message Validation**: Source elements within the inbound communication business object or raw message used for validation of asset inbound communications of this type
- **Communication Data Mapping**: Information used to populate communication business object elements from information within the communication's raw message
- **Attachment Suppression**: Substitution text used in place of large data attachments in the completion event list on the inbound communication. Used when messages received from an external system contain attachment data that are very large.

Refer to Defining Asset Inbound Communication Types for more information about setting up asset inbound communication types.

Defining Asset Inbound Communication Types

You use the **Communication Type** portal to display and maintain asset inbound communication types.

Refer to Understanding Asset Inbound Communication Types for more information.

You can access the portal by selecting **Admin**, then **Asset Management**, then **Asset Inbound Communication Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Communication Type List**: This zone lists all asset inbound communication type records. Broadcast a record to display the details of the selected record.
- **Communication Type**: This zone provides information about the selected communication type, such as attachment suppression, communication data mapping, message validation and exception handling.

Asset Types

Understanding Asset Types

Asset types define the attributes for assets and components of a certain type including information such as the valid location types and service history types for the assets and whether or not such assets can have attached components.

Asset types are defined by:

- **Component Relationship**: A flag that indicates if components can be installed on assets or components of this type
- **Asset Business Object**: The business object used for assets or components based on this asset type
- **Valid Location Types**: A list of location types where assets or components of this type can be located
- **Valid Attached To Asset Types**: A list of valid asset types where components of this type can be attached (applicable to components only)
- **Valid Service History Types**: A list of valid service history types that can apply for assets or components of this type

Refer to Defining Asset Types for more information about setting up asset types.

Asset Classes

Assets are defined by classes, each of which is based on a specific business object, and determines the type of asset: an asset or a component.

The table below lists the asset classes and the business object for each.

Asset Class	Business Object
Asset	W1-AssetType (Operational Device Asset Type)
Component	W1-ComponentType (Operational Device Component Type)

Defining Asset Types

You use the **Asset Type** portal to display and maintain asset types.

Refer to Understanding Asset Types for more information.

You can access the portal by selecting **Admin**, then **Asset Management**, then **Asset Type**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Asset Type** : This zone provides information about the asset type.
- **Asset Type Statistics**: This zone displays statistics for the asset type.

Questions

Understanding Questions

Questions are used to guide workers through corrective and preventive inspection processes as part of an asset's service history.

Questions are associated to a service history type which is then associated to an asset type.

During a corrective or inspection process, workers can be guided through a series of questions assessing the condition of the asset. The answers can be "weighted", which when calculated, gives an overall score as to the condition of the asset. The calculation is generally a sum (answer points * question weight), but can vary depending on the service history type. The total inspection points are then normalized to an overall grade or score. The condition rating and confidence rating are also calculated and the asset is updated with the appropriate values.

A question's answers can be defined as check boxes, multiple choice, or free-form text.

Refer to Defining Questions for more information about defining questions.

Defining Questions

You use the **Question** portal to display and maintain questions.

Refer to Understanding Questions for more information.

You can access the portal by selecting **Admin**, then **Asset Management**, then **Question**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Question**: This zone provides information about the question.
- **Service History List**: This zone displays a list of service history types that reference the question.

Service History Types

Understanding Service History Types

Service history types define the main attributes for service history of a certain type. Types might include maintenance, inspection, service, test, measurement, and so on.

Service history types are defined by the following:

- **Service History Business Object**: The business object used to create service history records of this type.
- **Category**: The category to which service history records of this type belong. The base package includes the following categories:
  - Downtime
  - Failure
  - Inspection
  - Maintenance
  - Repair

- **Subclass**: The specific subclass of the service history type, based on the **Service History Business Object**. Oracle Utilities Customer To Meter supports two subclasses of service history types:
  - Operational Device Service History
  - Questionnaire Service History

  Each service history type subclass includes other specific configuration information, outlined below.

Operational Device Service History Types

Operational Device Service history types are also defined by the following:

- **Service Schedules**: One or more template work orders and time intervals that should be used to schedule work order generation for service history records of this type

Questionnaire Service History Types

Questionnaire Service history types are also defined by the following:

- **Follow-up Actions**: One or more actions intended as “follow-up” actions for service histories of the service history type, such as work orders or work order activities, users can configure those follow-up actions on the service history type. Note that if a regulatory compliance requirement is associated with a follow-up action, a compliance type must be specified. For example, a Service History Type of “Grade 2 Gas Leak”, might include a follow-up action of “Recheck” which could be configured with a “30 day” Compliance Type and a follow-up action of “Repair” which could be configured with a “1 year” Compliance Type.
- **Stop Processing Service History Type**: One or more service history types that should stop the processing of **Follow-ip Actions**. For example, certain problem that needs follow up actions was reported. Then another service history, which indicates the problem has been fixed or its severity has changed, is recorded. Businesses might want to stop/cancel the follow up actions based on the original problem/severity.
- **Questions**: A list of one or more questions used to guide inspection of an asset. Questions are defined in the **Questions** portal.

Defining Service History Types

You use the **Service History Type** portal to display and maintain service history types.

Refer to Understanding Service History Types for more information.

You can access the portal by selecting **Admin**, then **Asset Management**, then **Service History Type**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Service History Type**: This zone provides information about the service history type.
- **Questionnaire Tree**: This zone displays the service history types’ questionnaire, based on the questions defined for the service history type. Applicable for Questionnaire service history types only.
- **Unlink Asset Types**: This zone lists the asset types currently linked to the service history type. To unlink one or more asset types, check the corresponding checkboxes and click **Remove**.
- **Link Asset Types**: This zone is used to search for and link asset types to the service history type. To link new asset types,

  1. Enter search criteria in the **Asset Type Description** field.
  2. Click **Search**.
  3. Select the asset types to link by checking the corresponding checkboxes.

4. Click **Link**.

Configuration Types

Understanding Configuration Types

Configuration types define the information required for various asset configurations with other assets, components, firmware, and other data.

Configurations are referenced on specifications which are then referenced on assets and components to indicate the best way that these entities can work with the other entities that they are connected to in the field.

Configuration types are defined by the following:

- **Configuration Class**: The class for configurations of this type
- **Configuration Category**: The category of configurations of this type
- **Configuration Business Object**: The business object used for configurations of this type

Configuration Classes

In general configuration classes define the types of connections that are being configured. Each configuration class is based on a specific business object, which is the business object used for configurations of each class and type.

The table below lists the configuration classes and the business object for each.

Configuration Class	Business Object
Single Asset	W1-SingleAssetConfig (Single Asset Configuration)
Asset/Component	W1-AssetCompConfig (Asset/Component Configuration)
Asset to Asset	W1-AssetToAssetCompConfig (Asset to Asset/Component Configuration)
Asset/Component to Asset	W1-AssetCompToAssetConfig (Asset/Component to Asset Configuration)
Asset to Asset/Component	W1-AssetToAssetConfig (Asset to Asset Configuration)
Asset/Component to Asset/Component	W1-AssetCompToAssetCompConfig (Asset/Component to Asset/Component Config)

Defining Configuration Types

You use the **Configuration Type** portal to display and maintain configuration types.

Refer to Understanding Configuration Types for more information.

You can access the portal by selecting **Admin**,, then **Asset Management**, then **Configuration Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration Type**: This zone lists all configuration type records. Broadcast a record to display the details of the selected record.

- **Configuration Type** : This zone provides information about the selected configuration type.

Configurations

Understanding Configurations

Configurations document and manage valid setups for assets and components. This includes a specification for each asset or component in the configuration as well as the valid value for each parameter, such as the firmware version of each asset.

Configurations are defined by the following:

- **Configuration Type**: The configuration type upon which the configuration is based
- **Status**: The current status of the configuration
- **Configuration Detail**: Specific information related to the configuration such as the assets and components involved, the firmware, characteristics, identifiers and other important information

Defining Configurations

You use the **Configuration** portal to display and maintain configurations.

Refer to Understanding Configurations for more information.

You can access the portal by selecting **Admin**, then **Configuration Report**, then **Configuration**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration**: This zone provides information about the selected configuration.
- **Configuration Attachments**: This zone lists any attachments defined for the configuration.

Configuration Reports

Understanding Configuration Reports

Configuration reports provide a summary of any currently installed configurations.

These reports show whether the configuration in an installation is valid or invalid, the identifiers and values being used, and display all of the assets/components and other elements that make up the installed configuration.

Configuration reports are defined by the following:

- **Reporting Mode**: The mode in which the report is run (One Time or Recurring)
- **Status**: The current status of the report (Pending or Submitted)
- **Report Criteria**: Specific information used to establish the filter criteria when generating the report, such as the assets and components involved, the firmware, characteristics, identifiers and other important information.

- **Report Result**: Specific information for each configuration, including the number of configurations, the assets and components involved, and the specifications and firmware versions of each. If the configuration is invalid, the **Configuration** column is blank in the report results. Users can navigate to specific valid configurations to review and edit the setup as needed. In addition, users can generate a work order to correct, reconfigure, or otherwise maintain any of the configurations by clicking **Generate Activity**.

Base Package Configuration Reports

The base package includes a number of configuration reports, each based on one of the support configuration classes (see Understanding Configuration Types.

The table below lists the configuration reports and the business object for each.

Configuration Report	Business Object
Single Asset Configuration Report	W1-SingleAssetCfgRpt
Asset/Component Configuration Report	W1-AssetCompCfgRpt
Asset to Asset Configuration Report	W1-AssetToAssetCfgRpt
Asset/Component to Asset Configuration Report	W1-AssetCompToAssetCfgRpt
Asset to Asset/Component Configuration Report	W1-AssetToAssetCompCfgRpt
Asset/Component to Asset/Component Configuration Report	W1-AssetCompToAssetCompCfgRpt

Defining Configuration Reports

You use the **Configuration Report** portal to display and maintain configuration reports.

Refer to Understanding Configuration Reports for more information.

You can access the portal by selecting **Admin**, then **Configuration Report**, then **Configuration Report**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration Report**: This zone provides information about the selected configuration report.

Chapter 8

Defining Location Options

NOTE: The asset location functionality described in this section is available only to cloud implementations using Oracle Utilities Meter Solution Cloud Service.

Asset Location Types

Understanding Asset Location Types

Asset location types define the details of asset or component locations of a certain type. Valid types of asset locations can include service points, other locations for installed assets, or any other location as defined by your organization.

Asset location types are defined by the following:

- **Class**: The general class for asset locations of this type, such as commercial or residential, a shop or warehouse, etc.
- **Location Business Object**: The business object used for asset locations of this type
- **Number of Assets Allowed**: A flag that indicates whether or not asset locations of this type can accommodate a single or multiple assets
- **Service Type**: The type of service (electric, water, etc.) that asset locations of this type support
- **Allow Related Work Locations**: A flag that indicates whether asset locations of this type should be allowed to be connected to a work location
- **Valid Parent Location/Organization Types**: One or more valid parent locations and parent organizations that can be referenced on asset locations of this type. Used primarily for reporting.
- **Valid Communicate-To Location Types**: One or more asset location types to which asset locations of this type can communicate. The **Tree View** tab of any asset or location will display other related assets with a visual indication of their relationship.

Defining Asset Location Types

You use the **Asset Location Type** portal is used to display and maintain asset location types.

Refer to Understanding Asset Location Types for more information.

You can access the portal by selecting **Admin**, **Location**, then **Asset Location Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Asset Location Type List**: This zone lists all asset location type records. Broadcast a record to display the details of the selected record.
- **Asset Location Type**: This zone provides information about the selected asset location type.

Organization Types

Understanding Organization Types

Organization types define the main attributes for organizations of a certain type.

Organization types are defined by the following:

- **Class**: The general class organization. Parent organizations have a class of “Parent”.
- **Organization Business Object**: The business object used for organizations of this type
- **Valid Parent Organization Types**: One or more valid parent organizations that can be referenced on organizations of this type. Used primarily for reporting.

Defining Organization Types

You use the **Organization Type** portal is used to display and maintain organization types.

Refer to Understanding Organization Types for more information.

You can access the portal by selecting **Admin**, then **Location**, then **Organization Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Organization Type List**: This zone lists all organization type records. Broadcast a record to display the details of the selected record.
- **Organization Type**: This zone provides information about the selected organization type.

Organizations

Understanding Organizations

Organizations are used for categorization and reporting for locations.

For example, your reporting may include a hierarchy that groups locations by a structure of parent and child organizations.

Organizations are defined by the following:

- **Organization Type**: The organization's type. Understanding Organization Types for more information.
- **Organization Disposition**: The current status of the organization (Active or Inactive)
- **Parent Organization**: The parent organization, if applicable.
- **Main Contact**: A contact for the organization

Defining Organizations

You use the **Organization** portal to display and maintain organizations.

Refer to Understanding Organizations for more information.

You can access the portal by selecting **Admin**, then **Location**, then **Organization**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Organization**: This zone provides information about the organization.
- **Child Locations / Organizations**: This zone lists any child locations and/or organizations for the current organization.

Out of Service Location Types

Understanding Out of Service Location Types

Out of service location types define attributes of types of locations where assets that are out of service are stored and maintained. Examples of out of service locations include warehouses, storerooms, and repair shops.

Out of service location types are defined by the following:

- **Class**: The general class (storeroom, meter shop, etc.) of out of service locations of this type
- **Location Business Object**: The business object used for out of service locations of this type
- **Valid Parent Location/Organization Types**: One or more valid parent locations and parent organizations that can be referenced on out of service locations of this type. Used primarily for reporting.

Defining Out of Service Location Types

You use the **Out of Service Type** portal is used to display and maintain out of service location types.

Refer to Understanding Out of Service Location Types for more information.

You can access the portal by selecting **Admin**, then **Location**, then **Out of Service Location Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Out of Service Location Type List**: This zone lists all out of service location type records. Broadcast a record to display the details of the selected record.
- **Out of Service Location Type**: This zone provides information about the selected out of service location type.

Out of Service Locations

Understanding Out of Service Locations

Out of service locations are physical locations where assets are stored or repaired such as a storeroom, warehouse, or service facility. When assets are changed to statuses such as “In Repair” or “Pending Disposition” the asset location is typically moved to an out of service location.

Out of service locations are defined by the following

- **Location Type**: The out of service location’s type. See Understanding Out of Service Location Types for more information.
- **Location Disposition**: The current status of the out of service location (Active or Inactive)
- **Parent Location/Organization**: The out of service locations’ parent location or organization (if applicable)
- **External ID**: An ID used to identify the out of service location in external systems
- **Address Information**: The physical address of the out of service location
- **Contacts**: The primary contact for the out of service location. See About Contacts and Maintaining Contacts for more information about creating and maintaining contacts used with out of service locations.
- **Field Information**: Additional information about the physical location, such as GPS coordinates

Defining Out of Service Locations

You use the **Out of Service Location** portal to display and maintain out of service locations.

Refer to Understanding Out of Service Locations for more information.

You can access the portal by selecting **Admin**, then **Location**, then **Out of Service Location**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Out of Service Location**: This zone provides information about the out of service location.
- **Current Activities**: This zone provides a summary of the activities that are currently associated to the location.
- **Asset Summary**: This zone provided statistics about the number of assets, by asset type, that are currently associated with the location.
- **Owned Attachments**: This zone lists attachments owned by the location.
- **Referenced Attachments**: This zone lists attachments related to the location.

> **NOTE:** See About Contacts and Maintaining Contacts for more information about creating and maintaining contacts used with out of service locations.

Chapter 9

Defining Device Options

Command Sets

Understanding Command Sets

Command sets are used to define a group of commands that are not eligible for a particular device. For instance, if Commission or Decommission commands should be considered ineligible for a particular device model, a command set that references the Device Commission and Device Commission business objects could be created and associated with that device model.

Command sets are specified for individual device models via the Manufacturer portal.

Individual devices of a particular model can be configured to override ineligibility if needed.

Configuring Command Sets

This portal is used to display and maintain a Command Set.

Refer to Understanding Command Sets for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Command Set**.

The following zones may appear as part of the portal's **Main** tab page:

- **Command Set List:** This zone lists all Command Set records. Broadcast a record to display the details of the selected record.
- **Command Set:** This zone provides information about the selected Command Set.

Manufacturers

Understanding Manufacturers

Manufacturers are the companies that makes devices.

A device's manufacturer is defined as an attribute of the device itself.

Each manufacturer can have zero or more models defined. Models for a single manufacturer can have diverse service types.

- Models can specify an **Exclude Command Set** that references commands that are not eligible for that model. Refer to Understanding Command Sets for more information.
- The **Device Command Set Override** field indicates if a command set defined by in **Exclude Command Set** field may be can be overridden and specified at the device.

Configuring Manufacturers

This portal is used to display and maintain a Manufacturers.

Refer to Understanding Manufacturers for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Manufacturer**.

The following zones may appear as part of the portal's **Main** tab page:

- **Manufacturer List:** This zone lists all Manufacturer records. Broadcast a record to display the details of the selected record. **Manufacturer:** This zone provides information about the selected Manufacturer.

Head End Systems

Understanding Head End Systems

Head end systems are systems that collect measurement data and meter events for eventual submission to the application. Many devices can communicate to the application through a single head-end system, but a utility may have numerous head-end systems through which they communicate with devices.

Head end systems utilize processing methods that specify the type of initial measurement data and device events to create for devices (and their related measuring components) based on measuring component type. Head end systems also utilize processing methods that specify how smart meter commands are processed.

Refer to Understanding Process Methods for more information about processing methods.

Head End Systems Impact Data Import and Export

Head end systems are configured to identify how a particular external system communicates data with Oracle Utilities Meter Data Management. This includes:

- The identifier type used to locate devices and measuring components. These are used both on import and export of data.

- The date/time format used in various data imports (i.e. whether or not the date/time format includes time zone information).

Please refer to the embedded help for more information about these fields.

Each head end system can be associated to an external system which is used to define the messages that can be sent to that service provider and how each message is sent.

Understanding SGG Adapter Configuration

This section describes how to use a head end system's SGG Adapter Configuration portal.

You can use the SGG Adapter Configuration portal to view configuration information and access configuration components for an SGG adapter head end system.

Note: This portal displays configuration information for head-end systems that reference an SGG Adapter Configuration Sheet extendable lookup..

To use the configuration information portal for an SGG adapter head end system:

Select **Admin > Device > Head End System**.

In the Head End system List zone, click the Broadcast icon for the head-end system you wish to view.

Click the SGG Adapter Configuration tab to view the configuration information.

The SGG Adapter Configuration portal contains the following zones:

- **SGG Adapter Configuration Tracker**: This zone displays the configuration details of the adapter, as defined by the SGG Adapter Configuration Sheet extendable lookup referenced on the head end system. The configuration details include:

The components required for usage and event processing and command processing. To view more details about the components, you can click the component name to go to the business object for the component. For example, you can click the business object "SSN - Connect or Disconnect" to go to the business object portal for the SSN - Connect or Disconnect business object.

Status messages describing the configuration status of components. The following table lists the status messages that may be displayed and the possible actions you can take:

If the status message is...	You can...
Set up this processing method	Click the processing role to set up the processing method.
This processing method has been configured	Click the processing role to view the configured processing method.
Update your processing method with a communication BO	Click the status message to set up the processing method.
Update your external system / outbound message type with an Message sender	Click the status message to go to the external system.
Update your processing method with an outbound message type	Click the status message to set up the processing method.
Add a value to get started	Click the status message to go to the extendable lookup.
Values Existing: (number)	Click the status message to go to the extendable lookup.
Add a communication type	Click the status message to go to the communication type portal.
Add a device event type	Click the status message to go to the device event type business object for the communication type. This message appears only for Echelon type adapters.
Master Configuration has been added	Click the status message to view the Master Configuration portal.
Add Master Configuration for this adapter	Click the status message to view the Master Configuration portal.

Upload Statistics Aggregators: This zone lists the IMD Upload Statistics Aggregator measuring components associated with the head-end system.

Configuring Head End Systems

This portal is used to display and maintain Head End Systems.

Refer to Understanding Head-End Systems for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Head End System**.

The following zones may appear as part of the portal's **Main** tab page:

- **Head End System List:** This zone lists all Head End System records. Broadcast a record to display the details of the selected record.
- **Head End System:** This zone provides information about the selected Head End System.
- **Processing Method List:** This zone provides the list of processing methods defined for the Head End System.
- **Translation Methods List:** This zone provides the list of translation methods defined for the Head End System.
- **Inbound BOs Send By Service Provider:** This zone lists inbound Business Objects that are sent by this Head End System. The identification is driven by the Business Object having a Business Object Option of type "Sent By Service Provider" that references the current Head End System.

Measuring Component Types

Understanding Measuring Component Types

Measuring component types define the most important properties of a measuring component.

Measuring component types define what a measuring component measures (KWH, temperature, etc.), how regularly it measures it, and whether it should be connected to a physical device, or if it's used as a scratchpad measuring component or an aggregator measuring component. Measuring component types also specify how the measuring component's final measurements should be stored, how the measuring component's user-defined values should be calculated, and specific rules governing validation, editing, and estimation (VEE) for measuring components of the type. In addition, measuring component types define display properties and valid attribute values for measuring components belonging to the type.

The following configurable items are available for most measuring component types:

- **Value Identifiers**: These store the values of UOM, TOU, and SQI that identify the measured amounts for measuring components of this type. Value identifiers specify the quantities stored on the measurement records for measuring components of this type. Please refer to the Measuring Component Type Value Identifiers topic later in this section for more information.
- **Valid VEE Groups**: These define the VEE groups considered valid for measuring components of this type. Each group supplied here will be available to be selected on measuring components of this type and act as an override to the Fallback VEE Groups.
- **Fallback VEE Groups**: These define default VEE groups for a specific VEE Group Role that can be used with all measuring components of this type. This alleviates the need to specify the same VEE groups on multiple measuring components of the same type. Changes made to these groups will automatically apply to all measuring components of this type unless they have specified their own VEE groups for that particular VEE Group Role. Each VEE group is designated a VEE group role that indicates when and how the VEE group is used (for initial load, manual override, estimation, etc.).

- **Eligible Profile Factors (interval only)**: These define the profile factors that are considered to be eligible for interval measuring components of this type. One profile factor can be identified as the default. The default profile factor will be automatically selected in system processing when a profile factor is required.
- **Valid Profile Factors for Conversion from Scalar to Interval (scalar only)**: These define the profile factors that are considered to be eligible for scalar measuring components of this type when converting scalar measurements to interval measurements. These profile factors are used to produce a curve of interval data from a scalar value. Without one of these factors defined scalar to interval conversion will use a flat line method (i.e. evenly divide the scalar value across the intervals). One profile factor can be identified as the default. The default profile factor will be automatically selected in system processing when a conversion profile factor is required.
- **Valid Scratchpad Measuring Component Types**: These define the scratchpad measuring component types considered valid for measuring components of this type.
- **Related Statistics Measuring Component Types**: These define the measuring component types that will be used to store statistical information about the historical usage of measuring components of this type. Please refer to Configuring Measuring Component Statistics for more information on how this list is used.
- **Display Properties**: Defines how measurement data for measuring components of this type is displayed, including:
  - *Display Configuration*: Details related to how measurements are displayed, including the 360 chart rendering method, number of hours of data to display, the maximum days to search for measurements, the default TOU map used, the TOU by Day Profile factor used, and default measurement condition.
  - *Event Bar Profiles*: The event bar profiles used when displaying measurement data for measuring components of this type. Event bar profiles are defined as values for the 360 View Event Bar Profile extendable lookup.
  - *Final Values Overlay Profiles*: The final values overlay profiles used when displaying measurement data for measuring components of this type. Final values overlay profiles are defined as values for the Final Values Overlay Profile extendable lookup.
  - *Measurement Conditions Not Shown on Chart*: The measurement conditions that should be omitted from rendering onto 360 Degree charts. Measurements whose conditions match these values will be rendered as gaps. For example, many 360 Degree charts use the condition "No Read - System" to represent the lack of a measurement, by adding this condition to this list it a gap will be rendered instead of a line with a 0 quantity measurement and a condition of "No Read - System".

When creating a measuring component type the following options are available:

Name	Details	Business Object
Interval Channel Type - Physical	Provides the configuration for a physical interval channel (e.g. interval size). This is recommended for measuring components that measure consumptive usage.	D1-IntervalChannelTypePhysical
Interval Channel Type - Scratchpad	Provides the configuration for a scratchpad interval measuring component.	D1-IntervalChannelTypeScratchp
Interval Channel Type - Physical Subtractive	Provides the configuration for a physical subtractive interval channel. In addition to standard interval configuration (e.g. interval size) it also provides additional subtractive specific configuration (e.g. rollover validation, estimate reevaluation, etc). This is recommended for measuring components that measure subtractive usage.	D1-IntrvlChanTypPhysSubtractiv
Register Type	Provides the configuration for a physical register that is manually read (e.g. rollover validation). These can be either consumptive or subtractive but are expected to be read infrequently (e.g. once a month).	D1-RegisterTypePhysical
Auto-Read Register Type	Provides the configuration for a physical register that is automatically read. In addition to the standard register configuration (e.g. rollover validation) it also provides details around the schedule of expected readings	D1-AutoReadRegisterType

	(e.g. first daily measurement time and expected hours between measurements). These can either be consumptive or subtractive but are expected to be read frequently (e.g. at least once per day). Note: these measuring component types can also be used to model interval data that is received with a large interval size (e.g. 24 hours).	
Aggregator Type	Provides the configuration for an aggregation measurement.	D2-AggregatorType

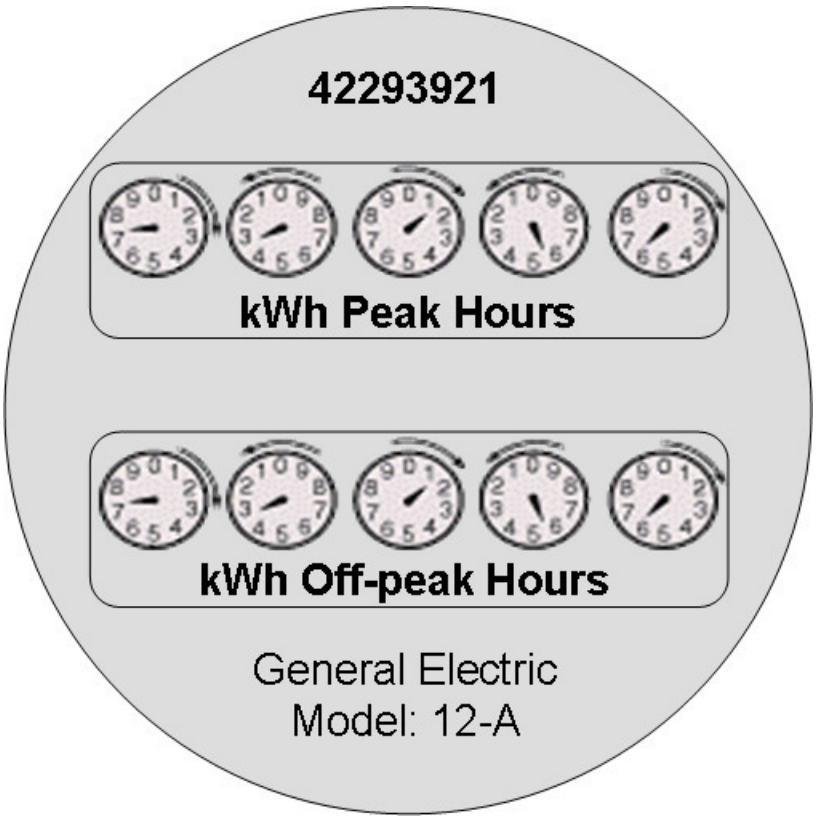
Refer to Configuring an Out-of-the-box Aggregation for more information about aggregation configuration

Measuring Component Type Value Identifiers

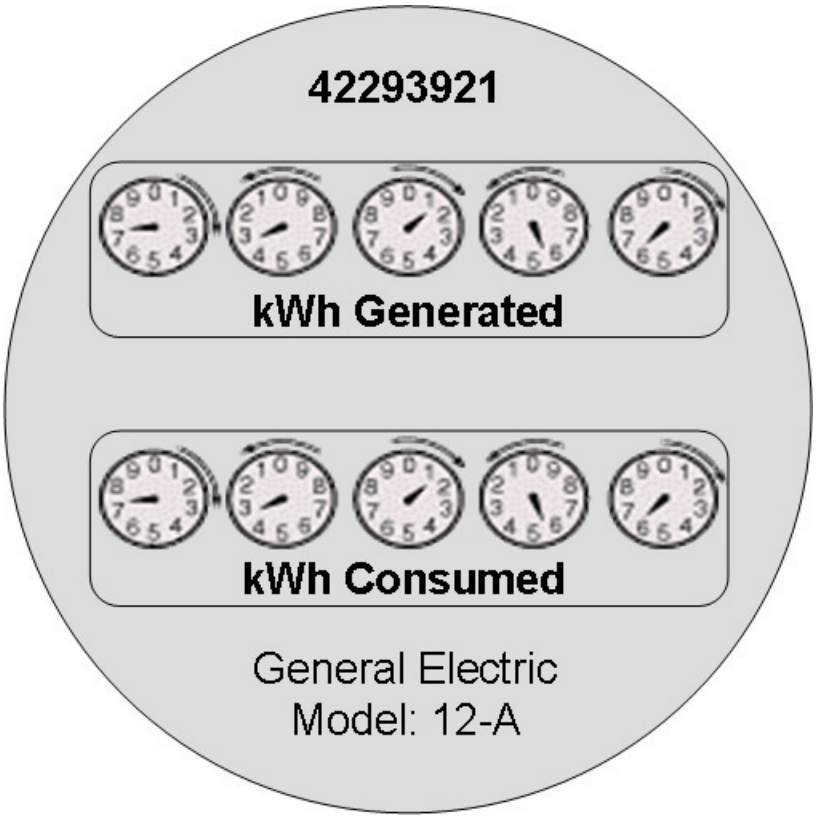
Measuring components are configured to measure specific types of quantities this is defined by the list of value identifiers on the measuring component type. Up to eleven value identifiers can be specified. The primary measured quantity should be identified using the Value Identifier Type of Measurement. An additional ten derived values can be computed based on the primary measurement, these are identified with the Value Identifier Type of Value 01 through Value 10. Each value identifier is constructed of:

- **Unit of Measure**: The unit of measure for the quantity being recorded. Examples include kilo-watt hours (kWh), kilo-watts (kW), therms, cubic feet (CCF), temperature (Fahrenheit or Celsius), etc. Refer to Understanding Units of Measure for more information.
- **Time of Use**: Modifiers for a given unit of measure that indicate a period of time during which a quantity has been used, such as On-Peak (meaning during a time when the greatest quantity of some consumable is being used), Off-Peak (meaning during a time when the least amount of some consumable is being used), etc. Refer to Understanding Time of Use for more information.
- **Service Quantity Identifiers**: Used to further distinguish between measured quantities that have identical UOM/TOU combinations, including situations in which the distinguishing identifier of a UOM is not accurately described as a TOU. Generally, SQI is only used when multiple measuring components measure the same thing, but in different ways. A meter that measures both generation KWH and consumption KWH could use SQIs to differentiate between the two. Refer to Understanding Service Quantity Identifiers for more information.
- **Value Derivation Algorithm**: Unlike UOM, TOU, and SQI this is not used in the identification of what is measured but rather is used to calculate a derived value based on the primary measurement. An algorithm from the list should be selected that contains the appropriate logic for calculating the derived value. This is applicable for those value identifiers with a Value Identifier Type of Value 01 through Value 10. More information on how derived values are calculated can be found in the Important Measuring Component Type System Events topic in this chapter. For more functional information about derived values please refer to About Final Measurements.

The combination of UOM, TOU and SQI define what a measuring component measures. TOU and SQI are optional, but UOM must be defined for all value identifiers. For example, consider a meter (as illustrated in the image below) with two measuring components, both measuring the same unit of measure (kWh), but each measuring component measures consumption in different time of use (TOU) periods (peak and off-peak).



Another example might be a meter that records both generated KWH and consumed KWH. This meter would be configured to measure both UOM and SQI.



A measurement is recorded each time a measuring component is read. This means that for a meter with two measuring components that is read once a month, two measurements, one for each measuring component, would be recorded each month.

Important Measuring Component Type System Events

The measuring component type supports several business object algorithm system events that relate to calculating the consumption for measuring components of that type:

- **Calculate Interval Consumption**: receives the interval list and performs any necessary calculations on that interval data to compute consumption. Since interval data is already received as consumption data algorithms for this system event are typically limited to application of the appropriate multipliers. Refer to the algorithm type Calculate Interval Consumption (D1-IN-CNSUMP) as an example.
- **Calculate Scalar Consumption**: receives information about the scalar reading and calculates the consumption as appropriate. Algorithms for this event typically support calculating the consumption using a stop and start reading or backing into a reading using consumption and a start reading. Much like the interval counterpart it will apply the appropriate multipliers. Refer to algorithm type Calculate Scalar Consumption (D1-SC-CNSUMP) as an example.
- **Calculate Subtractive Interval Consumption**: receives the interval list and supporting information (e.g. the start reading for the first interval) and either calculates the consumption by subtracting the interval's reading from the prior interval's reading or calculates the reading by adding the current interval's consumption to the prior interval's reading. Refer to algorithm type Calculate Subtractive Interval Consumption (D1-SIN-CNSUM) as an example.
- **Condition Mapping**: receives the details of a single subtractive interval along with the details for its start reading and computes the applicable final condition and reading condition. This is leveraged solely for subtractive interval measuring component types. Refer to algorithm type Subtractive Interval - Condition Mapping (D1-SIN-CNMAP) as an example.

These system events are typically called from within initial measurement processing during the initial stages of the initial measurement lifecycle (e.g. the Pre VEE status of most initial measurement business objects).

In addition to the measuring component type business object algorithms there is an additional system event provided on measuring component type itself:

- **Value Derivation**: receives details of an initial measurement and an associated final measurement. Using these inputs it can compute a value derived either from the primary measurement or one of the other derived values. Refer to algorithm type Derive a quantity using a formula (D1-DERIVAQTY) as an example.

Important Measuring Component System Events

The measuring component business object that is associated to a given measuring component type supports a special system event that is used in the periodic estimation process:

- **Periodic Estimation**: this system event will scan the measuring component's final measurement history to identify missing measurements and create either a To Do, or an estimation initial measurement, or both. More detail about this system event can be found by visiting the following algorithm types: Refer to algorithm type Create Interval IMD and To Do Based Upon Install History (D1-CIITBIH) and Auto-Read Scalar Periodic Estimation (D1-ARSPE) as an example.

Measuring Component Business Object Options

The device business object that is associated to a given device type plays an important role in how a device is processed in a system beyond defining the data associated to that device. Below are a list of business object options that are defined on the device business object and their impact on system processing:

- **Estimation Initial Measurement Data BO**: Is used to identify the appropriate estimation IMD to create.
- **Manual Override IMD BO**: Is used to identify the appropriate manual override IMD to create.
- **System IMD BO**: Is used to identify the appropriate system IMD to create.
- **Measuring Component Consumption Function**: identifies a function that can be executed from any compatible measuring component 360 Degree zone. Each measuring component can support 0 to many of these functions.

- **Interval Initial Measurement Function**: identifies a function that can be executed from any compatible initial measurement zone. Each measuring component can support 0 to many of these functions.

More detail about these options can be found by visiting a measuring component business object and inspecting the business object options.

Configuring Measuring Component Types

This portal is used to display and maintain a Measuring Component.

Refer to Understanding Measuring Component Types for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Measuring Component**.

The following zones may appear as part of the portal's **Main** tab page:

- **Measuring Component Type:** This zone provides information about the selected Measuring Component Type.

Measuring Component Comparison Types

Understanding Measuring Component Comparison Types

Measuring component comparison types define the details by which measuring component data can be compared to determine the days that most closely resemble a specific day being evaluated. For example, measuring component comparison types can be used on measuring components that store weather data so that comparisons can be made based on temperature. These are used when performing “proxy day” estimations, allowing the application to search for days that are similar to historical data (for instance, the same date from the previous month or previous year) to use as the basis for estimation.

Measuring component comparison types use the following parameters:

- A specific metric to be calculated to enable data comparisons. The “Calculate Comparison Coefficient - Pearson Correlation Coefficient” algorithm, included with the base package can be used to calculate a coefficient based on the linear correlation between two data sets (specifically between weather data stored on a measuring component and on the comparison object).
- The specific object to be compared. This can be either the same measuring component (to compare against historical data for the same measuring component), or a factor profile (to compare against profile data).
- The date range for the comparison period, including the start of the comparison period (current date, same date last year, or both), the time each day through which the comparison will be calculated, and the number of days prior to and after the start date that will be compared with the comparison object.
- Average Difference Eligibility: An average value either above or below the average for the current date that will be eligible for comparison. For example, assume the current date’s average is 75, and the prior day’s average is 72. If the Average Difference Eligibility is set to 3 then the prior day value of 72 will be eligible for comparison to the current day average of 75.
- The maximum number of resulting calculations that should be stored as measuring component comparable periods.
- A template that defines which days are considered like each day of the week (for example, weekdays are like other weekdays, weekend days are like other weekend days). This template is defined as a record for the “Like Day Eligibility Template” extendable lookup.
- Work calendars used to determine if specific data should be compared to other weekend days or to the same date in prior years, and which dates, if any, should be excluded from historical data used for comparison.

Configuring Measuring Component Comparison Types

Refer to Understanding Measuring Component Comparison Types for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Measuring Component Comparison Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Measuring Component Comparison Type List**: This zone lists all measuring component comparison type records. Broadcast a record to display the details of the selected record.
- **Measuring Component Comparison Type**: This zone displays details for the selected measuring component comparison type.

Device Configuration Types

Understanding Device Configuration Types

Device Configuration types define the measurements a given device can record. It should be noted that the device configuration type itself does not specify this information but rather the device configuration type will identify the valid list of measuring component types that can be referenced associated to a device.

When creating a new device configuration type there are the following options:

Name	Details	Business Object
Device Configuration Type	This represents configurations for physical meters and communication components. It identifies a list of valid measuring components types.	D1-DeviceConfigurationType
Item Configuration Type	This represents badged items that are installed at a particular service point. Given the nature of items there are no measuring component types associated to this type.	D1-ItemConfigurationType

Refer to Configuring Consumption Synchronization for more information on how a device configuration type can be set up to synchronize consumption between two related channels of measurement data.

Configuring Device Configuration Types

This portal is used to display and maintain a Device Configuration Type.

Refer to Understanding Device Configuration Types for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Device Configuration Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Device Configuration Type List:** This zone lists all Device Configuration Type records. Broadcast a record to display the details of the selected record.
- **Device Configuration Type:** This zone provides information about the selected Device Configuration Type.

Device Types

Understanding Device Types

Device types define information about a class of devices, including properties that apply to all devices of a type. Properties defined for a device type can be overridden for an individual device

Specifically, device types provide information about:

- How a device can be configured by defining the valid list of device configuration types
- How a device communicates by defining a fallback head end system and a list of valid head end systems.
- How a device records measurement data through the fallback incoming data shift which plays an important role in daylight savings processing. Refer to Daylight Savings Time Support for more information.
- The consumption profile of an item: the unit of measure consumed and method of calculation (profile or straight line)
- Whether the device is attached to a service point or meter (used with items and device communication equipment)

When creating a new device type there are the following options:

Name	Details	Business Object
Smart Meter Device Type	These devices measure consumption for a given service point, support remote commands, and remote collection of measurement and event data. These devices are associated to a head end system.	D1-SmartMeterType
Manual Meter Device Type	These devices measure consumption for a given service point but require a visit to the service point to collect the measurement data. They are not associated to head end systems.	D1-ManualMeterType
Item Device Type	These devices represent various items that consume consumption at a constant rate which are not directly metered. As consumption is calculated rather than measured so there is no need for an association to a head end system. Devices of this type can either be created individually for "badged" items or can be listed directly on a service point for "unbadged items." These devices may also be configured as equipment that can be attached to a service point.	D1-ItemType
Communications Component Device Type	These devices provide remote collection of consumption data measured by a manual meter on which they are installed. These devices are associated to a head end system.	D1-CommunicationCompMeterType
Device Communication Equipment Type	These devices provide remote collection of consumption data measured by a manual meter to which they are attached. Devices of this sort are sometimes referred to as Encoding Receiver/Transmitter (ERT) modules, or communication modules. These devices are associated to a head end system.	D1-DvcCommEquipmentType

Device Business Object Options Drive Functionality

The device business object that is associated to a given device type plays an important role in how a device is processed in a system beyond defining the data associated to that device. Below is a list of business object options that are defined on the device business object and their impact on system processing:

- **Device Category**: This defines the type of device and correlates to the core device types that are supported (i.e. smart meter, AMR meter, manual meter, item, or communication component). This value is used by smart meter commands, service order management, and other system processes to make processing decisions.
- **Install Event BO**: This identifies the appropriate install event business object to use when this device is being installed at a service point. Refer to About Install Events for more information about install events.
- **Valid Command Request BO**: This identifies the commands that are valid for any device of this type. This option can be repeated for as many commands as the device supports. Note: the combination of the device and service provider (aka head end) define the true list of available commands by device.
- More detail about these options can be found by visiting a device business object and inspecting the business object options.

Device Type - Service Quantity

For "badged" and "unbadged" items consumption is not directly measured. Instead, for each item type, the average daily consumption amount is provided. The commodity represented by the daily consumption amount is defined by the item type's unit of measure.

The average daily consumption amount can be effective dated over time to support changes in the consumption profile. This is used in conjunction with the item's calculation method to derive the consumed amount per interval.

Refer to Multiple Time Zone Support for additional information on how device type service quantities are impacted by multiple time zones.

Configuring Device Types

This portal is used to display and maintain a Device Type.

Refer to Understanding Device Types for more information.

You can access the portal by selecting **Admin**, then **Device**, then **Device Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Device Type List:** This zone lists all Device Type records. Broadcast a record to display the details of the selected record.
- **Device Type:** This zone provides information about the selected Device Type.
- **Device Type Service Quantity:** This zone displays the list of effective dated service quantities supported by the selected Device Type. This applies only to Items.

Chapter 10

Defining Device Installation Options

Service Point Types

Understanding Service Point Types

Service point types define a specific type of point at which service is delivered.

Specifically, service point types define how the application manages many aspects of the service point's behavior. A service point type may have one or more valid device types defined that limit the types of devices that can be installed at service points of this type.

Service point types are defined by the following:

- **Service Point Business Object**: the business object used by service points of this type
- **Service Point Category**: the types of devices that can be installed at service points of this type. Valid values include:
  - **Meter**: Indicates that a single meter can be installed at service points of this type.
  - **Item**: Indicates that a single "badged" item can be installed at service points of this type.
  - **Multi-Item**: Indicates that one or more "unbadged" items can be installed at service points of this type.
- **Valid Device Types**: the device types considered valid for service points of this type.
- **Valid Field Task Types**: the field task types considered valid for service points of this type. The field task types available when creating field activities can be based on the field task types listed here via the **Restrict Field Task Type by Service Point Type** option on the Service Order Management Master Configuration.
- **Usage Snapshot Configuration**: the configuration(s) to be used to take weekly or monthly usage snapshot(s)
- **Unreported Usage Snapshot Configuration**: the configuration(s) to be used to take weekly or monthly unreported usage snapshot(s)

Configuring Service Point Types

This portal is used to display and maintain a Service Point Type.

Refer to Understanding Service Point Types for more information.

You can access the portal by selecting **Admin**, then **Device Installation**, then **Service Point Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Service Point Type List:** This zone lists all Service Point Type records. Broadcast a record to display the details of the selected record.
- **Service Point Type:** This zone provides information about the selected Service Point Type.

Service Point Quantity Types

Understanding Service Point Quantity Types

Service point quantity types define types of quantities that can be stored for a service point. These quantity types are often infrequently calculated descriptors of a given customer and how they relate, at an aggregate level, to either the customer base as a whole or their particular rate class. For example, a customer quantity may be used to store a scaling factor that describes how a given customer's usage compares to the profiled usage for their rate class.

In a settlement implementation, service point quantity types can be used to define values calculated monthly, annually, etc., such as annual peak load contribution (PLC).

Service quantity types use the following parameters:

- The service type (electric, gas, water, etc.) for service point quantities of this type
- Quantity identifiers related to the current service point quantity type (used to provide shorthand descriptions of the various types of values measured by service point quantities of this type)

Configuring Service Point Quantity Types

Refer to Understanding Service Point Quantity Types for more information.

You can access the portal by selecting **Admin**, then **Device Installation**, then **Service Point Quantity Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Service Point Quantity Type List**: This zone lists all service point quantity type records. Broadcast a record to display the details of the selected record.
- **Service Point Quantity Type**: This zone displays details for the selected service point quantity type.

Measurement Cycles

Understanding Measurement Cycles

Measurement cycles define the schedule for manual meter reading of devices at Service Points in that cycle. Measurement cycles can have one or more associated routes used to collect measurements.

Measurement cycles can also be configured to define when to create usage transactions for Usage Subscriptions associated to Service Points in the cycle.

For a deeper functional understanding, refer to the About Route Management section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Measurement Cycles

This portal is used to display and maintain a Measurement Cycle.

Refer to Understanding Measurement Cycles for more information.

You can access the portal by selecting **Admin**, then **Device Installation**, then **Measurement Cycle**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Measurement Cycle List:** This zone lists all Measurement Cycle records. Broadcast a record to display the details of the selected record.
- **Measurement Cycle Type:** This zone provides information about the selected Measurement Cycle.
- **Measurement Cycle Route List:** This zone lists the measurement cycle routes related to the measurement cycle.
- **Measurement Cycle Schedule List:** This zone lists the measurement cycle schedules related to the current measurement cycle.

Measurement Cycle and Bill Determinants

The system can be configured to periodically push bill determinants to subscribing systems. In this case, measurement cycles can be configured to define when to create usage transactions for Usage Subscriptions associated to Service Points in the cycle. Even Service Points whose meters are read automatically may reference measurement cycles.

Creating bill determinants (by creating a usage transaction) is performed by an algorithm on the "Complete" state of the SP/Measurement Cycle Schedule Route business object (similar to creating activities as described above).

When the Pending SP/Measurement Cycle Schedule Route records are processed by the D1-PSPSR batch, rather than create a handheld download activity, the algorithm can create a usage transaction (usage transactions are transactions that cause bill determinants to be calculated for the Service Point's Usage Subscription).

If the implementation needs to both manually read the meter and push bill determinants, both algorithms would be plugged in on the SP/Measurement Cycle Schedule Route business object.

Measurement cycle processing is managed by the following three batch processes:

- **Create Pending Measurement Cycle Schedule Routes** (D1-CMCS): This batch process creates Schedule Routes for Measurement Cycle Schedules whose schedule selection date is on or before the batch business date. This process is used if routes have the same schedule each month, quarter, etc. This process simply copies the routes from the Measurement Cycle to the Measurement Cycle Schedule on/after the scheduled selection date.

- **Create Pending SP / Measurement Cycle Schedule Route Records** (D1-CSPSR): This batch process creates a "SP/ Measurement Cycle Schedule Route" transaction for every Service Point in the Measurement Cycle Schedule Route that is ready for processing.
- **Process Pending SP / Measurement Cycle Schedule Route Records** (D1-PSPSR): This batch process transitions the Pending "SP/Measurement Cycle Schedule Route" transactions to their Complete state. Custom algorithms can be configured to do any additional necessary work, such as creating a "Meter Read Download" activity. This custom algorithm would be configured as an Enter algorithm on the "Complete" state of the SP/ Measurement Cycle Schedule Route business object.

For a deeper functional understanding, refer to the About Route Management section of the *Oracle Utilities Meter Solution Business User Guide*.

Measurement Cycle Schedules

Understanding Measurement Cycle Schedules

Measurement cycle schedules define the dates on which devices are scheduled to be read for a given measurement cycle and the routes used to collect measurements for the measurement cycle.

For a deeper functional understanding, refer to the About Route Management section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Measurement Cycle Schedules

This portal is used to display and maintain a Measurement Cycle Schedule.

Refer to Understanding Measurement Cycle Schedules and Understanding Measurement Cycles for more information.

You can access the portal by selecting **Admin**, then **Device Installation**, then **Measurement Cycle Schedule**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Measurement Cycle Schedule Query:** This zone allows you to query for Measurement Cycle Schedules based on Measurement Cycle and select the desired record.
- **Measurement Cycle Schedule:** This zone provides information about the selected Measurement Cycle Schedule.

Chapter 11

Defining Measurement Options

Initial Measurement Data

Configuring the Initial Measurement Algorithms

The behavior of initial measurement data processing can be adjusted by updating parameters in several key algorithms in the lifecycle.

NOTE: This is not a list of all algorithms and algorithm parameters that can be customized in the initial measurement processing lifecycle. Rather, it is a selection of some of the more impactful parameters.

Initial Measurement Data Seeder Algorithms

- Derive Service Provider and Measuring Component (D1-DER-SPRMC):
  - Enabling a standalone measuring component search allows initial measurement data to be processed without a device identifier. This is useful when profile or temperature data is loaded through the IMD Seeder and a match must be made on measuring component identifiers alone.
  - If measuring component identifiers can be repeated across measuring components for the same device then the error for duplicate measuring components being found can be turned off. This is useful for certain head end systems that will have multiple channels with the same channel identifier. When a duplicate is found the search will fall back on other means for identifier the correct measuring component.
- Perform Date/Time Adjustments and Undercount/Overcount Check (D1-DODTTMADJ):
  - Undercount validation can be turned off completely or left enabled. When left enabled the automatic filling of gaps can be turned off separately.
  - The overcount check can be turned off.
  - The automatic adjustment of the individual interval date/times to an interval boundary can be turned on.

NOTE:
If your implementation receives initial measurement data with date/times that do not include an explicit time zone and the devices report date/times in standard time, you may need to add the following configuration:

- Navigate to **Admin** > **Integration** > **Message Option**
- In the **XAI compliance** option, ensure that the following text is provided:
  `xsd:strict:dstGapInStandardTime`

This will prevent a date/time reported in standard time that falls on the missing hour of the day Daylight Savings Time is entered from being misinterpreted.

Specific Initial Measurement Data Algorithms

- Normalize measurements (overwrite identical existing Measurements) (D1-NORM-IMD):
  - When the initial measurement data includes measurements that match exactly with the existing final measurements they can either be overwritten or skipped.
  - If logging of changes to final measurements is desired then it must be indicated as such in this algorithm. It is suggested to keep this turned off for high volume initial measurement data such as initial load. See Configuring Measurement Logging for more details.

NOTE: Refer to Configuring Consumption Synchronization for additional parameters that can be adjusted on specific initial measurement data algorithms.

Configuring Measurement Logging

There are two components to logging changes to final measurements:

- **Initial Measurement Finalization**: the "normalization" algorithm on a particular initial measurement data business object will determine how final measurements are updated. For performance reasons certain types of initial measurement data (e.g. initial load) are delivered with a final measurement update method that will skip any logging. This can be controlled by parameters on the algorithm that implements the algorithm type Normalize Measurements for Initial Measurement Data (D1-NORM-IMD). Specifically the "Create Measurement Log on Update (Y/N)" parameter should be set to "Y". As mentioned creating these logs has a performance impact and it is not recommended for use on initial load.
- **Measurement Business Object**: there must be an algorithm for the audit system event configured. The base package delivers the algorithm type Add Measurement Log Record (D1-AMSRMTLOG). It is this algorithm that actually records the log entries.

Logging User Updates to Manual Initial Measurements

By default, manual edits made by users of these zones are not logged on the Log tab. Logging of manual edits to manual initial measurements can be enabled by adding a logging algorithm on an appropriate lifecycle state of the manual initial measurement business objects. The Log User Transaction (D1-LOGUSRTRN) base package algorithm can be used for this. This Enter algorithm is designed to be defined on the Initial state of the manual initial measurement business objects, but it can also be defined on any (non-transitory) Interim or Final state as well.

To ensure logging of any or all manual edits made to manual initial measurements, this algorithm should be specified on any state in which users will make manual edits. This will most often be the Pending or VEE Ready states, but could also include the Error, Exception, or Finalized states.

CAUTION: When defining this algorithm, the user should exercise caution and determine if previous algorithms in the sequence within the state contain any form of transitioning logic that may inadvertently cause this algorithm to be bypassed.

Chapter 12

Defining Validation, Editing, and Estimation (VEE) Options

Exception Types

Understanding Exception Types

Exception Types define the groupings of exceptions for an IMD based on their functional similarity. This provides a way to define VEE Exceptions in a distinct enough way to understand the root issue that was generated from the VEE Rule.

For a deeper functional understanding of VEE, refer to the About VEE section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Exception Types

This portal is used to display and maintain an Exception Type.

Refer to Understanding Exception Types for more information.

You can access the portal by selecting **Admin**, then **VEE**, then **Exception Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Exception Type List**: displays all of the Exception Types so you can choose the one you want to display in more detail
- **Exception Type**: shows the specific configuration for the selected Exception Type

There are two different options to use when creating an Exception Type:

Name	Details	Business Object

VEE Exception	This will create a "normal" VEE Exception that is attached to an IMD for tracking of conditions triggered in the VEE process.	D1-VEEException
VEE Exception - Monitor Service Point	In addition to tracking the failure in the VEE process, this VEE Exception generates a Service Issue Monitor. This allows for cumulative tracking of VEE Exceptions that can be configured to result in a Service Investigative Order (field work) for the Service Point.	D2-VEEExceptionServiceMonitor

VEE Groups

Understanding VEE Groups

VEE groups are collections of VEE Rules that are applied to initial measurement data. During the VEE process, the system executes the VEE Rules defined in each VEE group. The rules within a VEE group are defined in a specific sequence, allowing control over the order in which the rules are executed.

VEE groups can be associated to a specific measuring component, or to a measuring component type (or both). VEE groups associated with a measuring component type are applied to all measuring components of that type, while those associated to a specific measuring component are applied only to that measuring component. VEE groups associated to a measuring component override those assigned to a measuring component type.

VEE groups can also be referenced by the Execute VEE Group VEE Rule.

For a deeper functional understanding of VEE, refer to the About VEE section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring VEE Groups

This portal is used to display and maintain a VEE Group.

Refer to Understanding VEE Groups for more information.

You can access the portal by selecting **Admin**, then **VEE**, then **VEE Group**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **VEE Group**: Defines basic information about VEE group
- **VEE Rules List**: lists the VEE Rules belonging to the group
- **Referencing VEE Rules List**: lists the VEE Rules that reference the group
- **Referencing VEE Group Factors List**: lists the VEE group factors that reference the group
- **Referencing Measuring Component Type List**: lists the measuring component types that reference the group
- **Referencing Measuring Component List**: lists the measuring components that reference the group

VEE Rules

Understanding VEE Rules

VEE Rules are standard and custom Validation and Estimation rules that perform checking and/or manipulation of initial measurement data. VEE Rules are created for a specific VEE group. For example, if you were configuring two VEE groups and both included a specific VEE Rule, you would need to create two instances of the VEE Rule, one for each group.

The specific validation and estimation processing performed on initial measurement data is defined in individual VEE Rules, each performing a specific set of targeted logic. The base product contains many VEE Rules you can use in your implementation, but you can also create your own custom VEE Rules.

Some VEE Rules generate VEE Exceptions if the initial measurement data fails the conditions specified for the rule. Other rules override measurements, changing measurement values as dictated by the rule's parameters. Some rules can both create exceptions and override the measurement as part of a single process. By convention, VEE Rules change the Post-VEE quantities of initial measurement data, but VEE Rules can change anything on an initial measurement.

Every VEE Rule has an effective period. Rules will only be applied if the initial measurement's start date is within the rule's effective period. For example, an Interval Spike Check rule with a Start Date of 11/15/2010 will only be applied if the start date of the initial measurement is on or after 11/15/2010.

This allows you to update the specifics of a rule without removing the previous version of the rule. For example, you might change the tolerance of an Interval Spike Check rule from 1.2 to 1.5 as of a certain date. However, for initial measurement data for the period prior to the change, you would want to use the tolerance for the original version of the rule (1.2) instead of the new tolerance (1.5).

On almost every VEE Rule, the failure of the rule results in a VEE Exception and the Exception Type for the failure can be configured on the rule. These Exception Types can also be set to a specific Exception Severity:

- **Information**: Used to highlight minor issues, but not sufficient to cause the initial measurement data to be put into the exception state. Exceptions of this category can be used to report on the frequency of interesting, but not fatal issues
- **Issues**: Used to report a problem that will prevent the initial measurement data from being finalized. Multiple "issue exceptions" can be created during VEE processing. If at least one issue exists after all rules have been applied, the initial measurement data is transitioned to the exception state
- **Terminate**: Used to report a severe issue that will cause the VEE process to stop and the initial measurement data to be transitioned immediately to the exception state. Only one terminate exception can be issued (as the first one causes VEE processing to stop on an initial measurement data).

For a deeper functional understanding of VEE, refer to the About VEE section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring VEE Rules

This portal is used to display and maintain a VEE Rule. Also, a list of the specific out-of-the-box rules is included below the instructions for using the portal.

Refer to Understanding VEE Rules, Understanding VEE Groups, and Understanding Exception Types for more information. For a deeper functional understanding of VEE, refer to the About VEE section of the *Oracle Utilities Meter Solution Business User Guide*.

You can access the portal by selecting **Admin**, then **VEE**, then VEE Rule. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **VEE Rule**: this zone displays all of the configuration items specific to this instance of a VEE Rule.

- **Eligibility Criteria List**: this zone displays any VEE Rule Eligibility Criteria that have been setup. These eligibility criteria determine conditionally whether the VEE Rule should be executed or not. Use the Add button to create a new eligibility criteria for the rule you're viewing.

Validation VEE Rules

Below is a list of the validation rules provided as part of base product. For more information on how each rule executes and can be configured, follow the link provided on the rule.

VEE Rule Name	Applicable Data Type(s)	Purpose
Consecutive Interval Check	Interval	This validation rule flags any combination of consecutive intervals within Initial Measurement Data based on the values of the data or the condition codes of the data.
Duplicate IMD Check	Interval or Scalar	This rule allows for a duplicate IMD to be flagged.
Dynamic Comparison Validation	Interval or Scalar	This powerful, flexible validation rule compares measurements to historical statistics for the related Service Point. The system will maintain statistics such as the following: sum, min, max, average, median, zero value count, outage count, and standard deviation. Then you define formulas (no programming required) for the comparison of current measurements to these statistics.
Ensure IMD Exists for Sibling MCs	Interval or Scalar	This rule validates that an IMD exists for all of the other measuring components associated to the same Device Configuration as the current measuring component, for the same period of time.
Final Measurement Replacement	Interval or Scalar	This validation rule allows you to define a variety of configuration options to decide if new data should replace existing measurements. The options include value change thresholds, percentage change thresholds, as well as condition code ranking.
High/Low Check	Interval or Scalar	This validation rule compares the total consumption of the current IMD to historical values. The comparison is normalized based on average daily usage (ADU). If the current IMD is too high or too low compared to historical data then an exception is thrown.
Inactive Measurement Check	Interval or Scalar	This validation rule flags any Initial Measurement Data received on a device that is either disconnected, uninstalled, and/or not connected to a Usage Subscription.
Interval Size Validation	Interval	This rule validates that the interval size (in seconds) supplied with the Initial Measurement is equal to the interval size defined on measuring component's type.

Interval Spike Check	Interval	This rule checks for spikes within an IMD and generates an exception if one is found.
Multiplier Check	Interval or Scalar	This rule validates that the register multiplier supplied with the IMD is equal to the multiplier stored on the measuring component. If not, an exception is created using the register multiplier exception type and severity configured on the rule.
Negative Consumption Check	Interval or Scalar	This rule flags any IMD where the total consumption is negative.
Prolonged Estimation Check	Interval or Scalar	This validation rule creates an alert when a device has been estimated for an extended period of time.
Raise Missing Quantity Exception	Interval	This rule flags any missing interval data.
Sum Check	Interval or Scalar	This rule is used to compare the difference between interval data to scalar data for a period of time, or between a set of TOU scalar reads to a "Sum" scalar reading.
Unit of Measure Check	Interval or Scalar	This rule checks the unit of measure (UOM) passed in with the Initial Measurement against the primary unit of measure configured on the measuring component's type.
Zero Consumption Check	Interval or Scalar	This rule checks if the total consumption for the IMD is zero. There is also a check for whether an outage occurred during the same time as the zero consumption to provide ways to avoid exceptions in that case.

Estimation VEE Rules

Below is a list of the estimation rules provided as part of base product. For more information on how each rule executes and can be configured, follow the link provided on the rule.

VEE Rule Name	Applicable Data Type(s)	Purpose
Interval Adjustment From Scalar	Interval	This rule performs adjustments to interval values based on the values from the associated scalar data on the same device.
Interval Averaging Estimation	Interval	This rule finds historical interval data from the same measuring component based on a variety of configuration options to use for estimating any missing data within an IMD.
Interval Create Estimation IMD for Gap	Interval	This estimation rule creates a new Estimation IMD to fill gaps between the interval data received from the Head End. ***Note:*** *this estimation rule acts differently from the other rules as it creates a new IMD rather than filling in missing values in an estimation IMD created from periodic estimations.*

Interval Interpolation Estimation	Interval	This rule attempts to interpolate gaps within an IMD using prior and subsequent intervals as starting points for linear interpolation.
Interval Profile Estimation	Interval	This rule estimates any missing interval data for the IMD based on a referenced Profile Factor.
Interval Proxy Day Estimation	Interval	This rule checks for days with similar temperature and uses the measurement data from those days as the basis of interval estimation for the in-flight initial measurement.
Scalar Calculation From Interval	Scalar	This rule performs adjustments to scalar values based on the values from the associated interval data on the same device.
Scalar Estimation	Scalar	This rule finds historical scalar data from the same measuring component based on a variety of configuration options to use for estimating any missing data within an IMD.
Scalar Profile Estimation	Scalar	This rule estimates any missing scalar data for the IMD based on a referenced Profile Factor.
Scalar Proration	Scalar	This rule prorates the value of a scalar reading that has two valid scalar readings on either side as boundaries. It will also take into account any related interval data within the same period to exclude from the calculation.
Subtractive Interval Adjustment Rule	Subtractive Interval Adjustment Rule	This rule performs adjustments to qualifying interval consumption values for subtractive interval measuring components based on an adjustment target calculated using a start and stop reading for the period that encapsulates the intervals to adjust.

Decision-Making VEE Rules

There are VEE Rules delivered as part of the base product that help with decision-making when executing VEE (listed below). For more information on how each rule executes and can be configured, follow the link provided on the rule.

VEE Rule Name	Purpose
Exception Handler	This rule allows for termination of the VEE process based on a configurable set of exceptions being present for the IMD. This rule also allows a unique To Do Type to be generated based on a group of exceptions.
Execute VEE Group	This rule performs a call to execute a separate VEE Group which includes execution of all VEE Rules within that group.
Successful Termination	This rule allows VEE to be successfully terminated based on a list of exceptions.

VEE Group Matrix (Factor)	This rule provides a way to choose different instances of a VEE Rule using a Factor. This factor leverages characteristics that are defined on Service Point, Device, or Measuring Component.

Validation VEE Rules

Consecutive Interval Check

Overview

This rule flags any combination of consecutive intervals within Initial Measurement Data based on the values of the data or the condition codes of the data. It can be used to find faulty meters that are reporting consecutive outage codes, zero measurements, or negative values. It can also be used by water utilities to identify leaks based on the interval never reaching zero.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-CONSINTRV Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

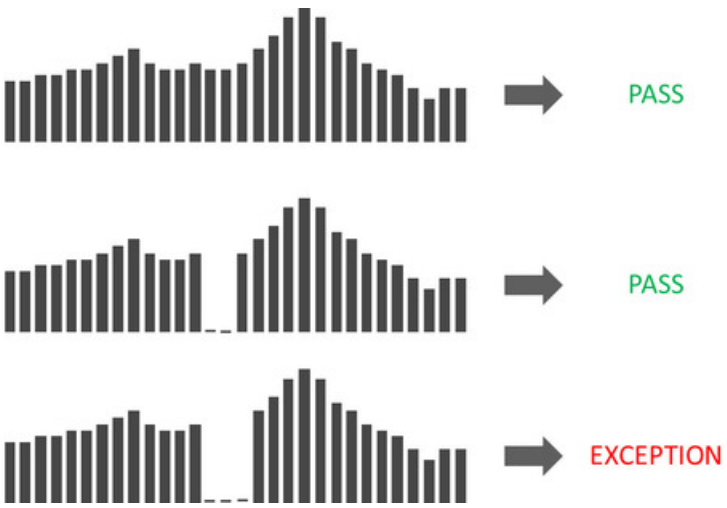
D2-ConsecutiveIntervalCheck

Example Scenarios

Below are some example scenarios that can be achieved based on configuration of this rule.

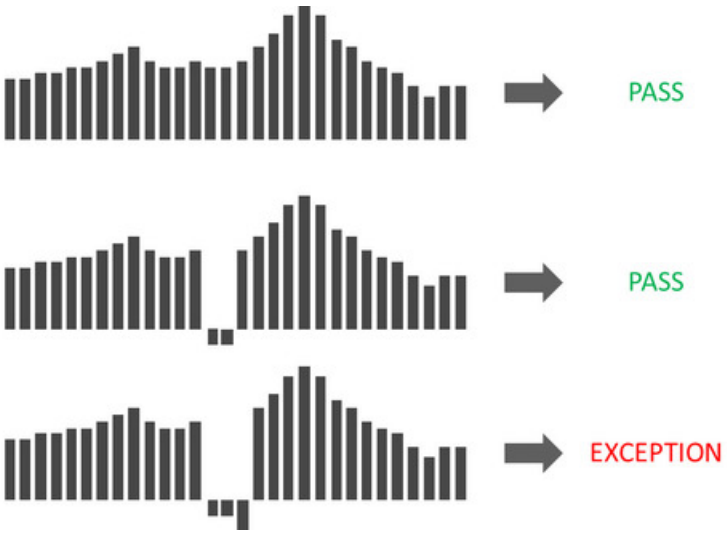
Scenario 1: Rule configured to fail for zero values

Parameter Name	Configuration Value
Check performed	Consecutive Measurement Values
Maximum consecutive hours	02:00:00
Look To Prior Measurements	No
Equation Type	Equal To
Value for Comparison	0
Bottom Range Condition	
Top Range Condition	



Scenario 2: Rule configured to fail for negative values

Parameter Name	Configuration Value
Check performed	Consecutive Measurement Values
Maximum consecutive hours	02:00:00
Look To Prior Measurements	No
Equation Type	Less Than
Value for Comparison	0
Bottom Range Condition	
Top Range Condition	

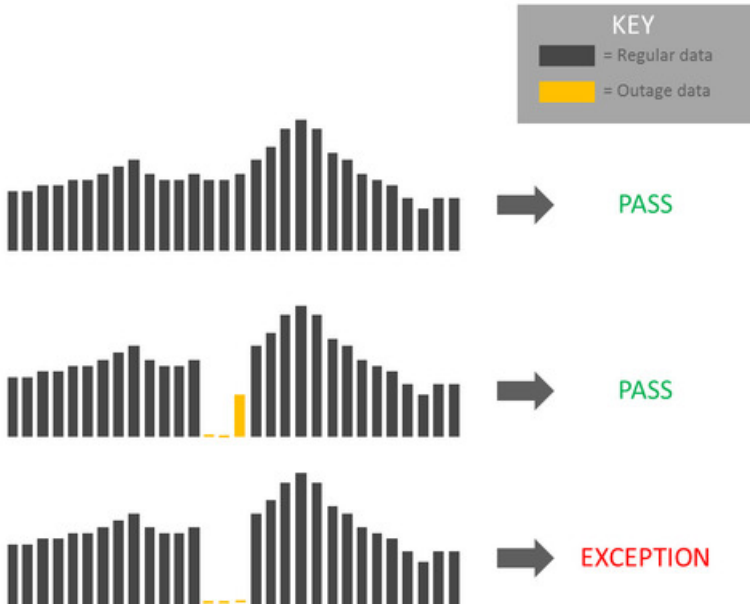


Scenario 3: Rule configured to fail for missing condition codes

Parameter Name	Configuration Value
Check performed	Consecutive Condition Values
Maximum consecutive hours	02:00:00
Look To Prior Measurements	No
Equation Type	
Value for Comparison	
Bottom Range Condition	Missing
Top Range Condition	Missing

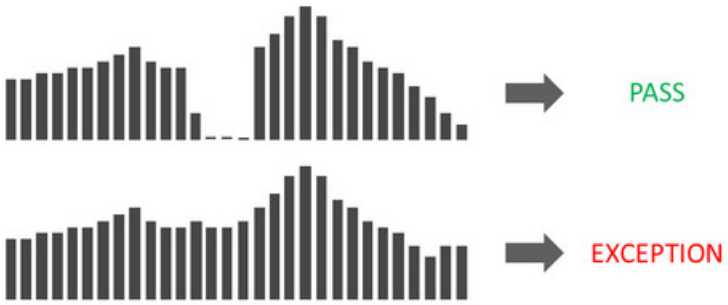
Scenario 4: Rule configured to fail for outage condition codes only if its zero

Parameter Name	Configuration Value
Check performed	Both
Maximum consecutive hours	02:00:00
Look To Prior Measurements	No
Equation Type	Equal To
Value for Comparison	0
Bottom Range Condition	Outage
Top Range Condition	Outage



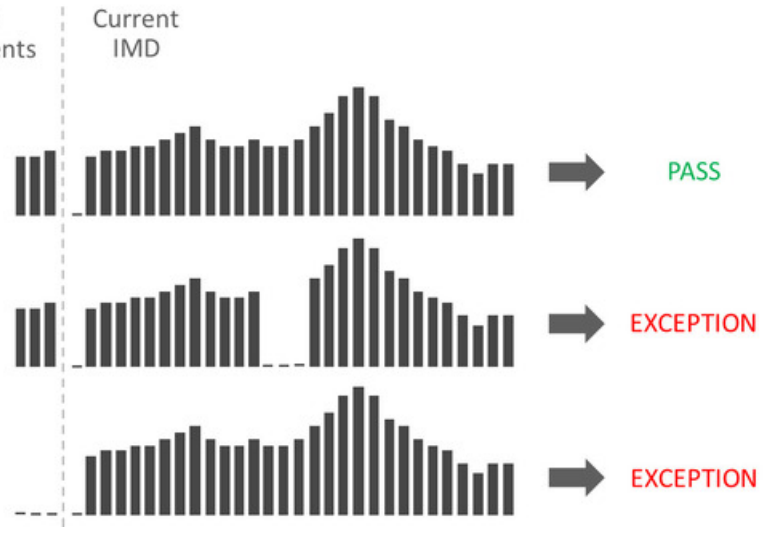
Scenario 5: Rule configured to fail if a long period of consecutive water values are greater than 0.1

Parameter Name	Configuration Value
Check performed	Consecutive Measurement Values
Maximum consecutive hours	20:00:00
Look To Prior Measurements When First Interval Matches Criteria	No
Equation Type	Greater Than
Value for Comparison	0.1
Bottom Range Condition	
Top Range Condition	



Scenario 6: Look back at prior measurements when the first interval matches criteria

Parameter Name	Configuration Value
Check performed	Consecutive Measurement Values
Maximum consecutive hours	02:00:00
Look To Prior Measurements	Yes
Equation Type	Equal To
Value for Comparison	0
Bottom Range Condition	
Top Range Condition	



Duplicate IMD Check

If the current Initial Measurement Data (IMD) being validated is determined to be a duplicate of an existing IMD for the same measuring component, this rule will produce a VEE exception of the type and severity configured on the VEE Rule. The algorithm logic looks for duplicate IMDs using the following criteria:

- associated to the same measuring component as the current IMD
- utilizes the same IMD business object as the current IMD (for example, Initial Load Scalar)
- references the same To Date/Time (ends on the same date) as the current IMD
- exists in a "Finalized" state
- the contents of pre-VEE are identical to the pre-VEE of the current IMD

If any IMDs are found that meet all of the above criteria, the current IMD is deemed to be a duplicate.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-DUPIMDCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D1-DuplicateIMDCheck

Dynamic Comparison Validation

This validation rule compares measurements to historical statistics for the related Service Point. The system will maintain statistics such as the following: sum, min, max, average, median, zero value count, outage count, and standard deviation. Setting up these Measuring Component Statistics is a prerequisite to using the rule.

Users can define formulas (no programming required) for the comparison of current measurements to these statistics. This is powerful rule will allow utilities to look for unusual usage patterns. For example:

- Lowest/highest usage ever
- Current usage is more than three standard deviations from the mean
- Detect unexpected zero usage
- Detect negative usage while ruling out known cases
- Abnormal usage
- Voltage threshold monitoring
-

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-DYNCOMCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

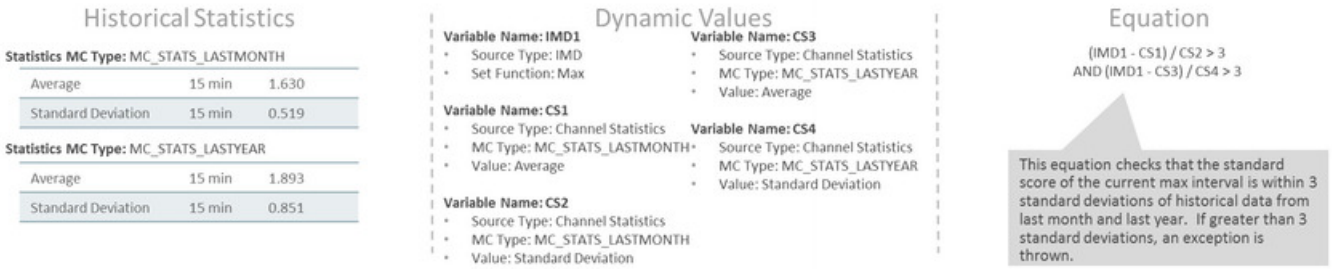
D2-DynCompValidation

Example Scenarios

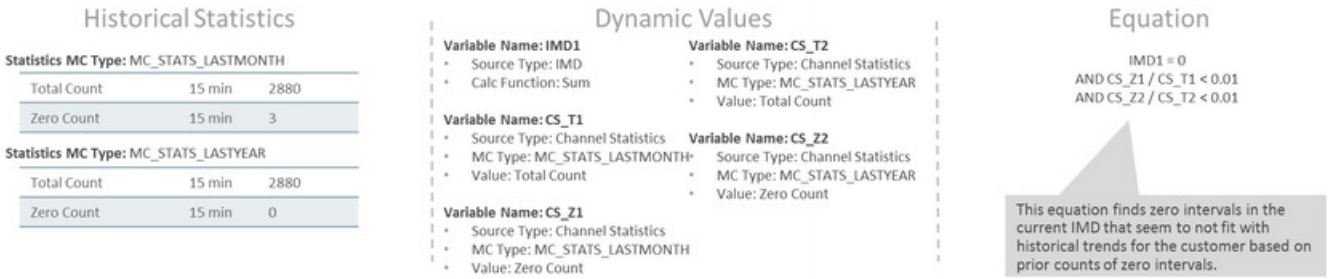
Below are some example scenarios that can be achieved based on configuration of this rule.

Example 1: Detect new "high water mark"

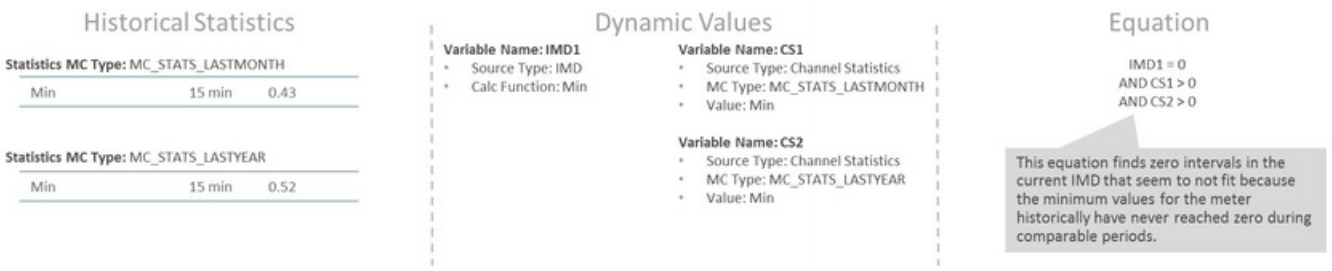
Example 2: High Usage



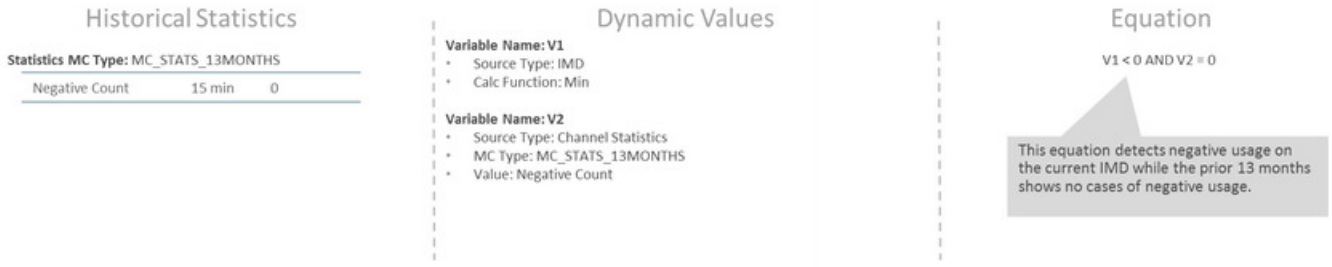
Example 3: Detect unexpected zero usage (method 1)



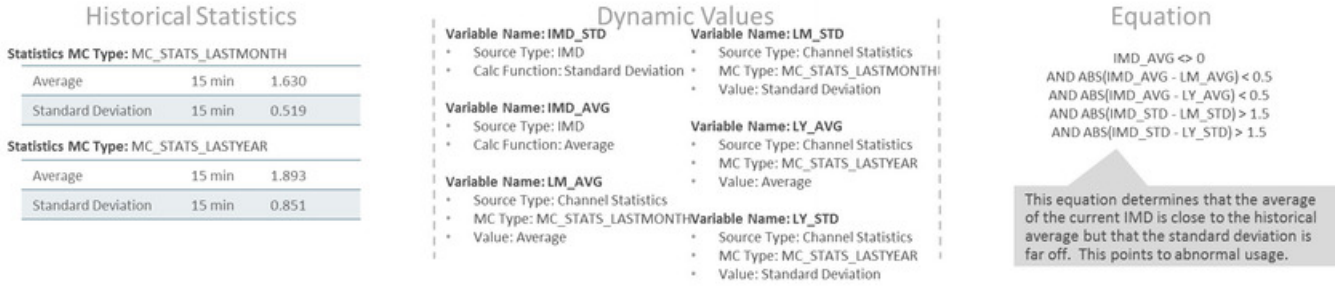
Example 4: Detect unexpected zero usage (method 2)



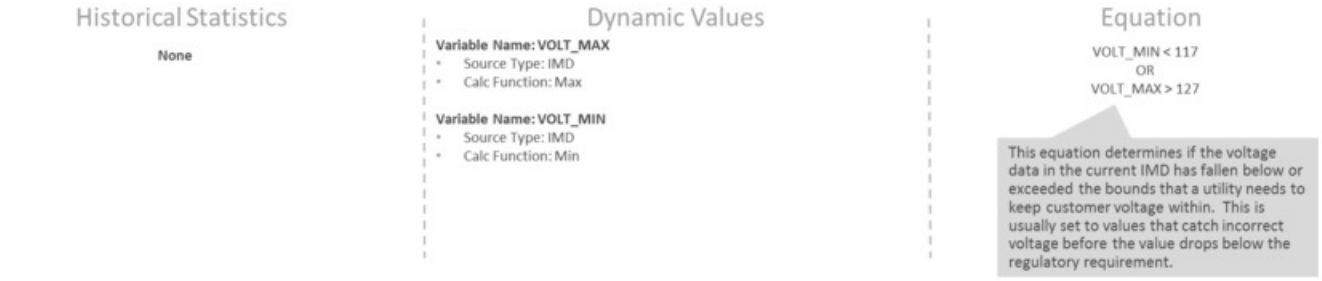
Example 5: Detect negative usage while ruling out known cases



Example 6: Abnormal usage



Example 7: Voltage threshold monitoring



Ensure IMD Exists for Sibling MCs

This rule validates that an IMD exist for all of the other measuring components associated to the same Device Configuration as the current measuring component, for the same period of time.

A check is also performed that all of the Initial Measurements have the same External Source ID (indicating that they all came from the same usage file).

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-ENSIMDMC Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-EnsureIMDExistsForSibling

Final Measurement Replacement

This validation rule allows you to define a variety of configuration options to decide if new scalar or interval data should replace existing measurements. The options include value change thresholds, percentage change thresholds, as well as condition code ranking. One common use for this rule is rejecting trivial measurement changes to prevent very small changes for a bill from being sent to a customer.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-VLMSRCOND Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

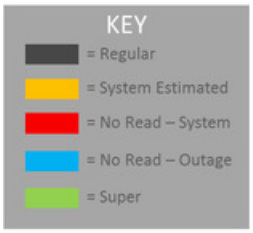
Business Object

D2-FinalMeasurementValidation

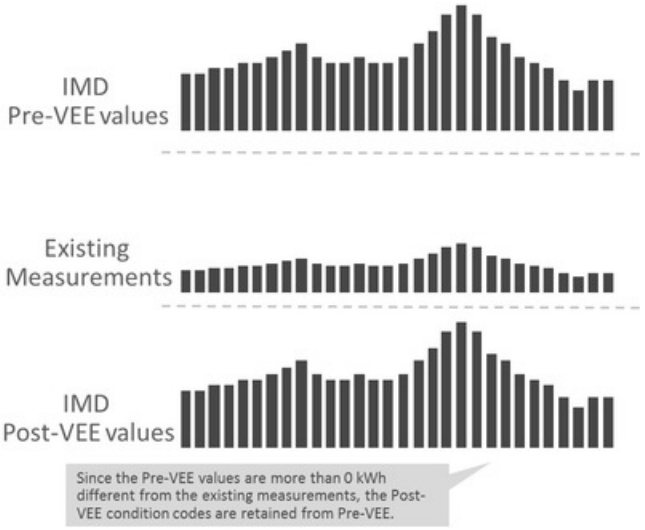
Example Scenarios

Below are some example scenarios that can be achieved based on configuration of this rule.

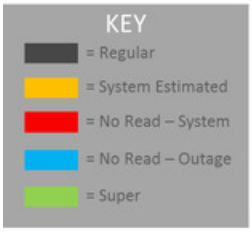
Example 1: Replacement of measurements based solely on value difference



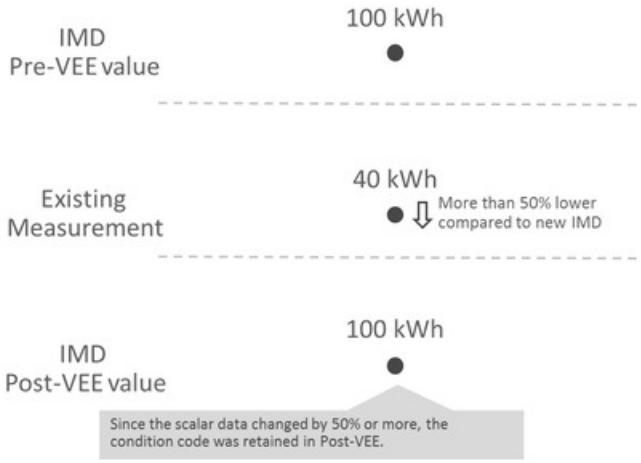
Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	Compare Value Change
Value Change Tolerance	0.1
Percentage Change Tolerance	
Condition Code Prioritization by Groups	



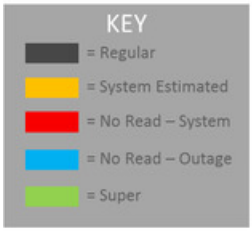
Example 2: Percentage change to replace measurement



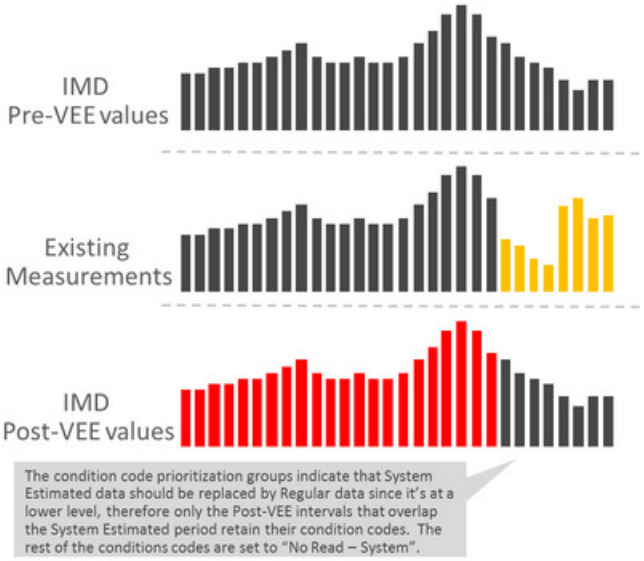
Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	Compare Percentage Change
Value Change Tolerance	
Percentage Change Tolerance	50
Condition Code Prioritization by Groups	



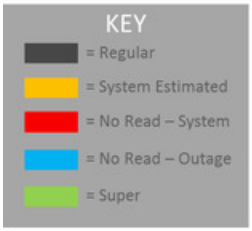
Example 3: Partial replacement of estimated data with regular data based on condition range prioritization



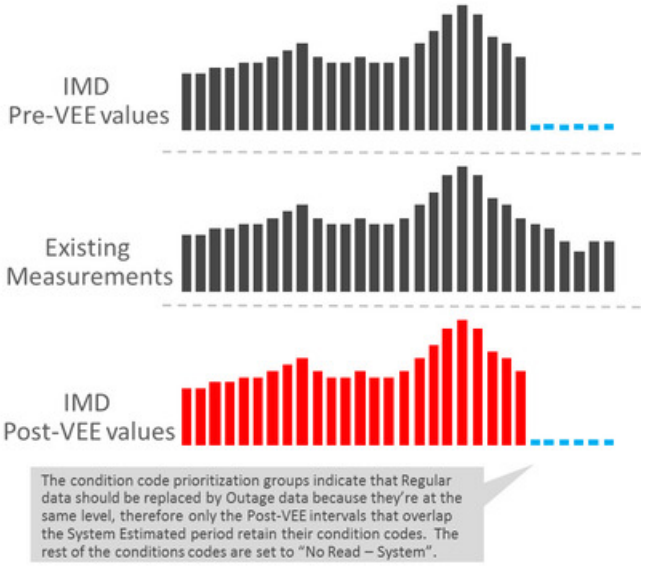
Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	Compare Value Change
Value Change Tolerance	0.1
Percentage Change Tolerance	
Condition Code Prioritization by Groups	10: Super 20: Regular, No Read – Outage 30: System Estimate 40: No Read – System



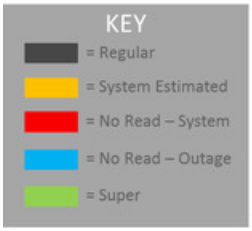
Example 4: Partial replacement of regular data with outage data based on condition range prioritization



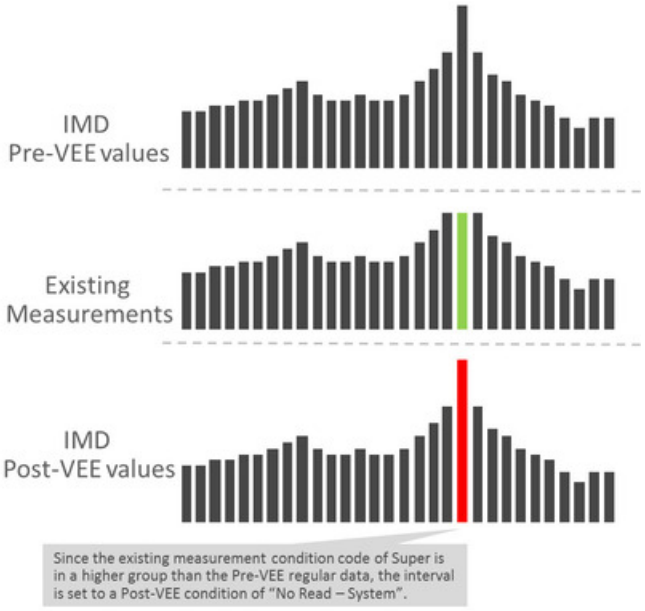
Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	Compare Value Change
Value Change Tolerance	0.1
Percentage Change Tolerance	
Condition Code Prioritization by Groups	10: Super 20: Regular, No Read – Outage 30: System Estimate 40: No Read – System



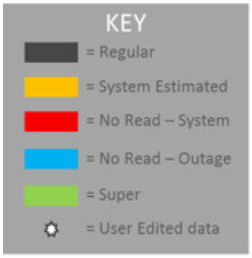
Example 5: Super not replaced by regular (retain a smoothed spike)



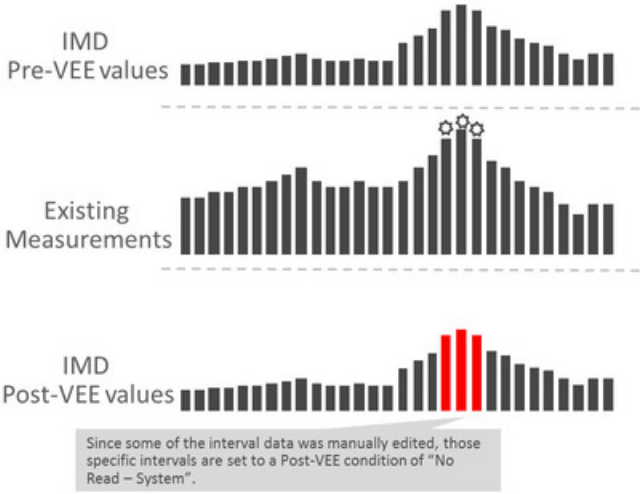
Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	
Value Change Tolerance	
Percentage Change Tolerance	
Condition Code Prioritization by Groups	10: Super 20: Regular, No Read – Outage 30: System Estimate 40: No Read – System



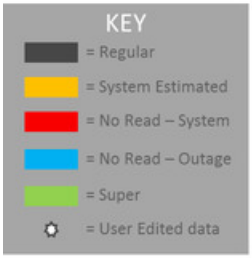
Example 6: Prevent replacement of manually edited data



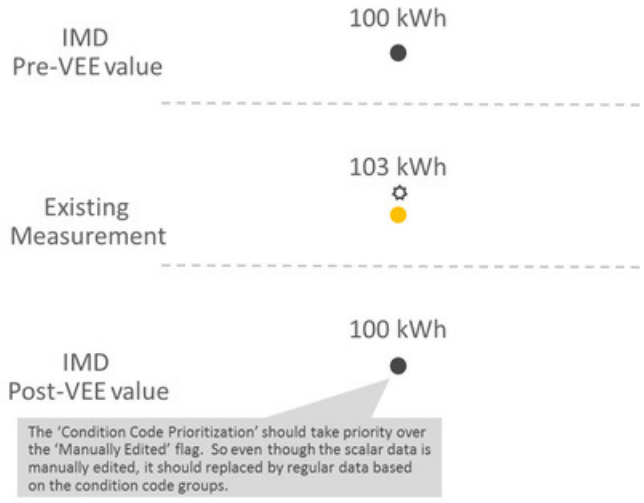
Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	No
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	
Value Change Tolerance	
Percentage Change Tolerance	
Condition Code Prioritization by Groups	



Example 7: Allow replacement of manually edited data if the condition group is lower than new data



Parameter Name	Configuration Value
Replacement Method	Reject Based on Configuration
Replacement Condition Value	No Read – System
Replace Manually Edited Data?	No
Use Early and Late MC Type Settings?	
Replacement Evaluation Type	
Value Change Tolerance	
Percentage Change Tolerance	
Condition Code Prioritization by Groups	10: Super 20: Regular, No Read – Outage 30: System Estimate 40: No Read – System



High/Low Check

This validation rule compares the total consumption of the current IMD to historical values. The comparison is normalized based on average daily usage (ADU). If the current IMD is too high or too low compared to historical data then an exception is thrown.

Numerous configuration options are provided including:

- A Historical Pre-Window and Post-Window that determines the number of days to check before and after the period being examined.

- Percentages to control how much of the historical data can be user edited, estimated, or non-normal data.
- What type of historical data to look at first
- What to do when an outage has occurred

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-HILO-CHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-VEERuleHighLowCheck

Inactive Measurement Check

This validation rule flags any Initial Measurement Data received on a device that is either disconnected, uninstalled, and/or not connected to a usage subscription.

- When the 'Check Measurements on Disconnected Device' flag is set to "Yes", the logic will compare the IMD measurements total to the threshold configured. If the total measurements are above the threshold, then logic compares the IMD dates to two things: 1) On/Off Dates related to the Install Event and 2) the Installation and Removal Dates of the Install Event. If the On or Off falls within the dates for an interval IMD, then the logic will only sum intervals that fall during the disconnected periods.
- When the 'Check for Uninstalled Device' flag is set to "Yes", the logic will check for a valid Install Event based on the IMD dates.
- When the 'Check for Missing Usage Subscription' flag is set to "Yes", the logic will check for an active Usage Subscription based on the IMD dates.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INACTVCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

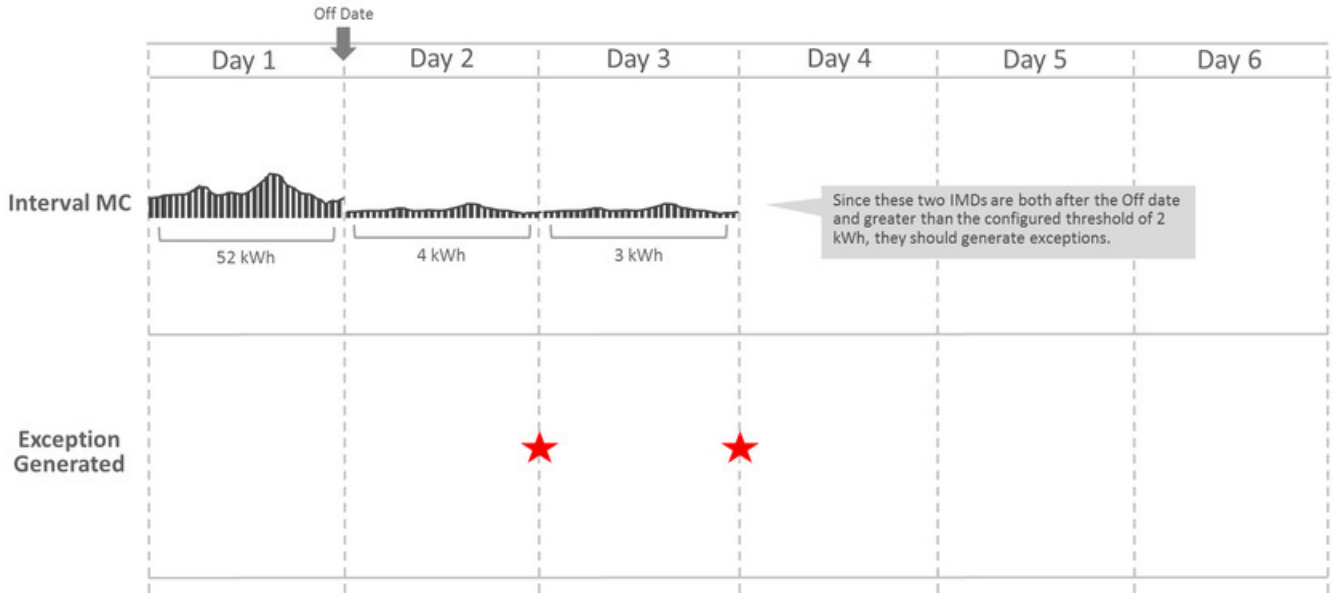
Business Object

D2-InactiveMeasurementCheck

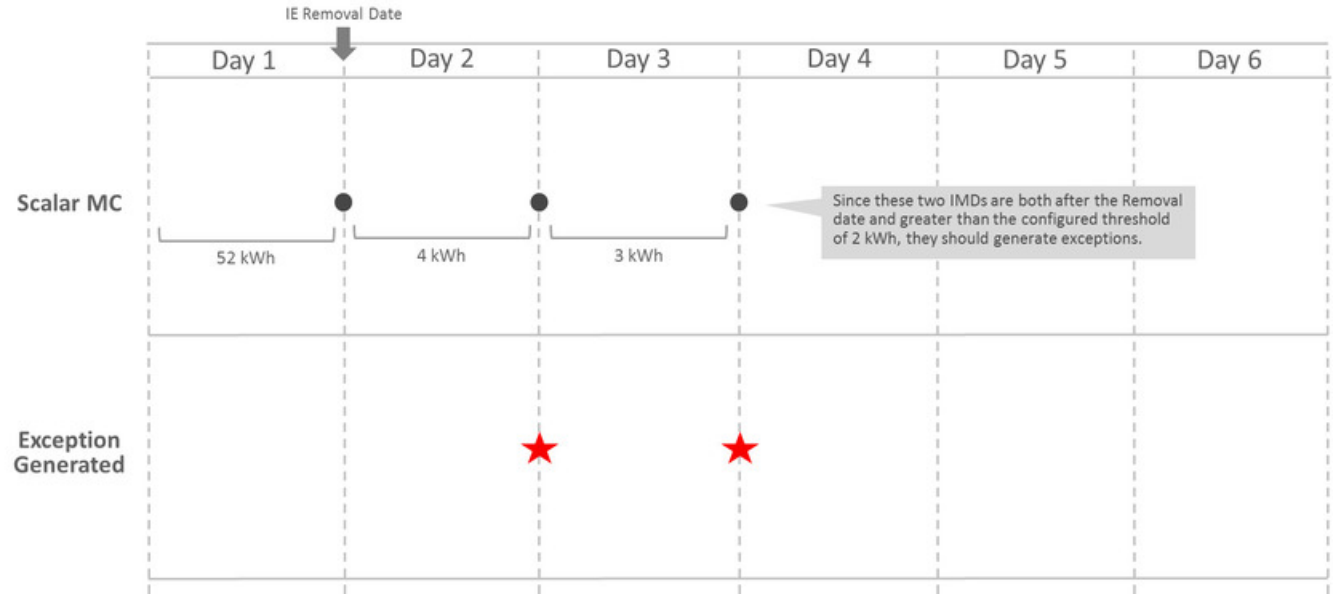
Example Scenarios

Below are some example scenarios that can be achieved based on configuration of this rule.

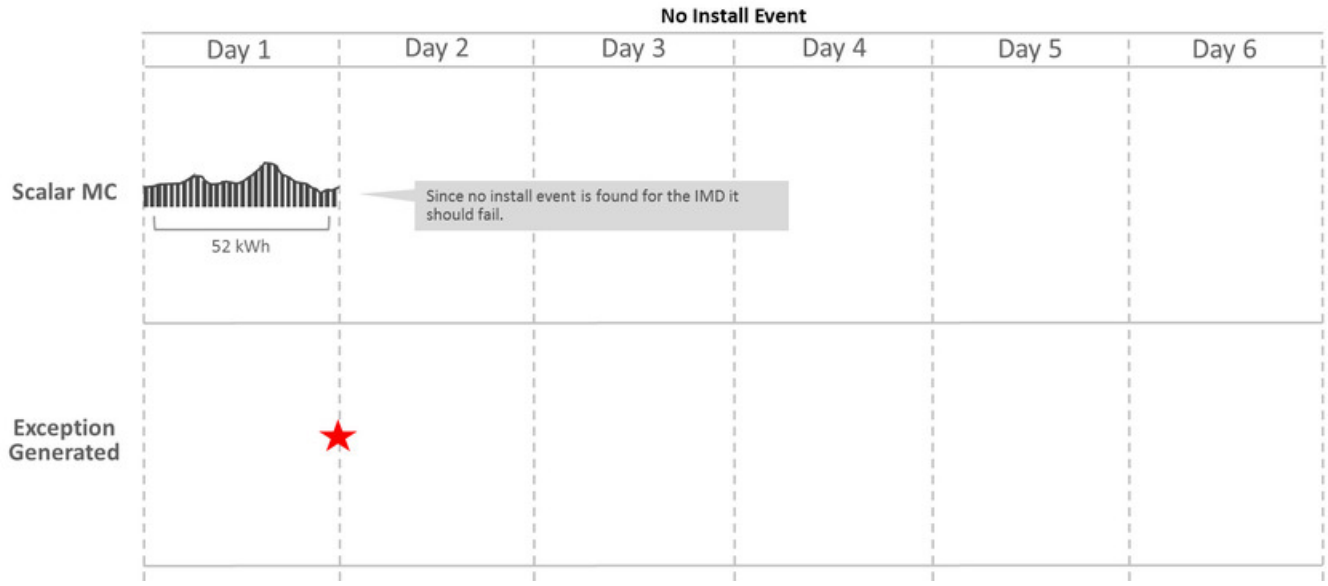
Example 1: Non-zero data during disconnected period



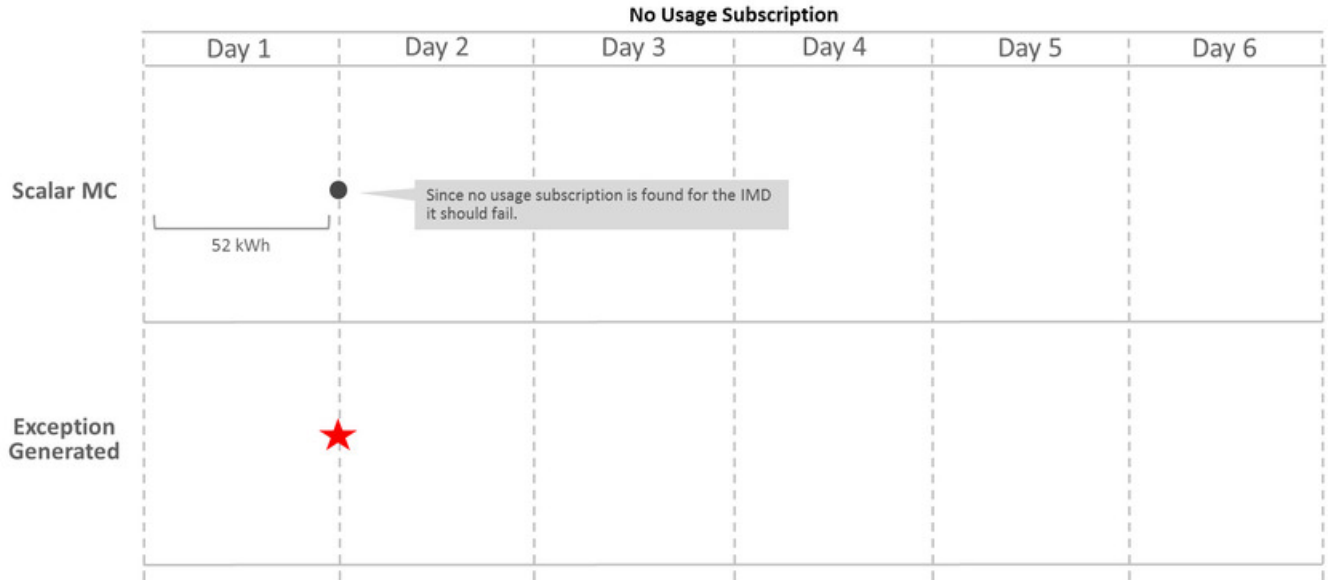
Example 2: Non-zero data after removal date with no new install



Example 3: Data received when device has no Install Event records



Example 4: Data received when not connected to a Usage Subscription record



Interval Size Validation

This rule validates that the interval size (in seconds) supplied with the Initial Measurement is equal to the interval size defined on measuring component's type.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTSIZVAL Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalSizeValidation

Interval Spike Check

This rule looks for spikes by taking the highest interval and the third-highest interval, and determining the percent difference between the two. If the percent difference is larger than the tolerance configured on the rule, the algorithm logs an exception of the type and severity configured on the rule.

The rule can be executed in one of two modes (as configured on the rule):

- The spike check is performed for every 24-hour chunk of data in the initial measurement data set
- The spike check is performed for the entire initial measurement data set

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTSPKCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalSpikeCheck

Multiplier Check

This rule validates that the register multiplier supplied with the Initial Measurement is equal to the multiplier stored on the measuring component.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-REGMULCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-RegisterMultiplierCheck

Negative Consumption Check

This rule checks if the total consumption of the IMD is less than zero. If the rule encounters negative consumption, an exception will be logged only if the related Measuring Component Type is not configured to "allow negative consumption".

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-NCON-CHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-NegativeConsumptionCheck

Prolonged Estimation Check

This validation rule can be used on either interval or scalar and will alert you when a device has been estimated for an extended period of time. The IMD must first fall within the condition code range configured on the rule for this to execute. Next the validation finds the 'Most Recent Non-Estimated Read Date Time' from the Measurement Component. If this date plus the 'Days of Estimation Allowable' from the VEE Rule is less than the IMD End Date then a Prolonged Estimation Exception is created.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-PROESTCHK Algorithm Type.

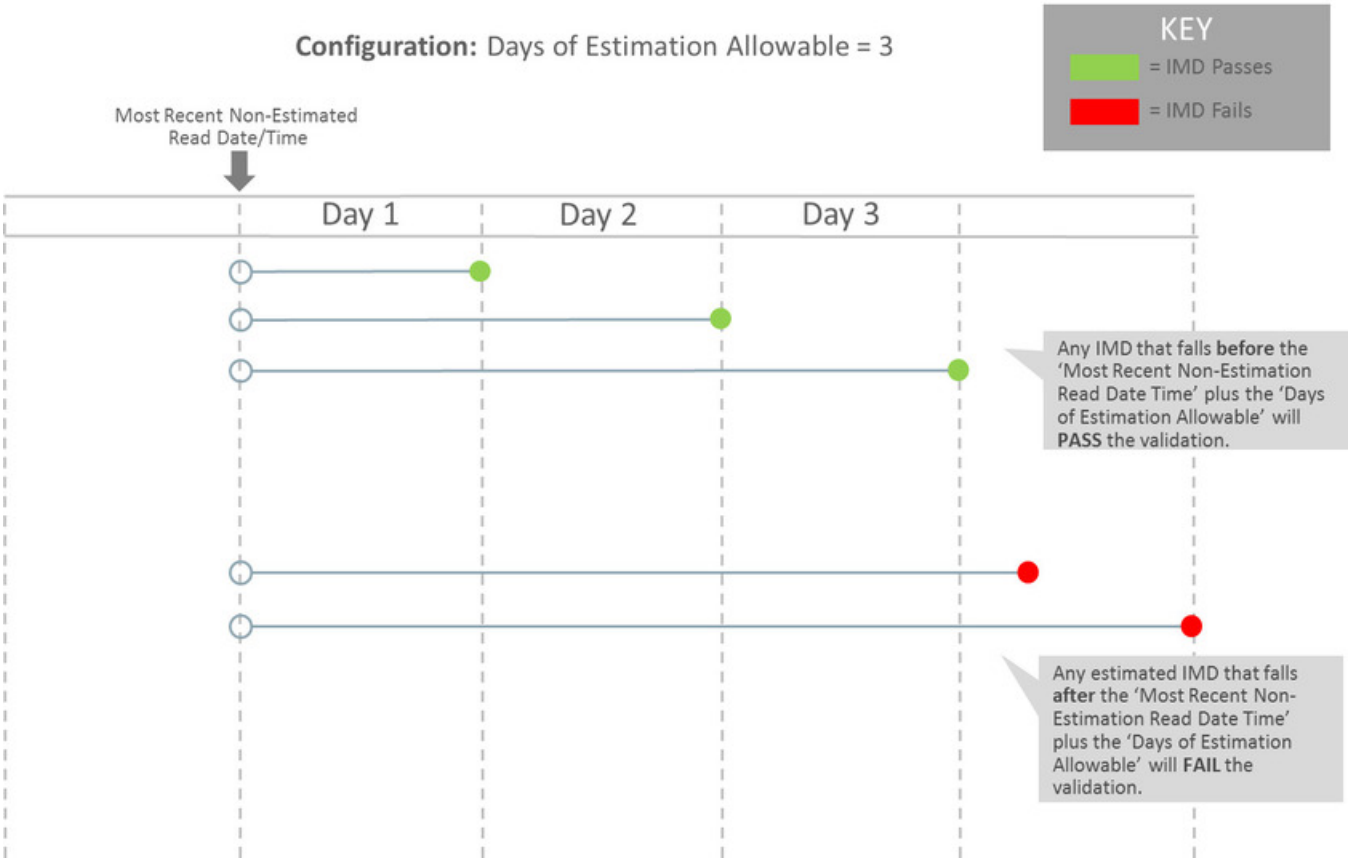
For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ProlongedEstimationCheck

Example Scenario

Below is an example scenario that can be achieved based on configuration of this rule.



Raise Missing Quantity Exception

This rule flags any missing interval data while providing a soft parameter on the algorithm to exclude a condition range if desired.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-RAIMISQTY Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-RaiseMissingQuantityExcp

Sum Check

This rule evaluates whether consumption for the current Initial Measurement Data is within a tolerance of the sum of the consumption during the same period for any measuring components related to the current one. If the values are not within the defined tolerance of each other, and exception is logged.

The rule can be used to evaluate consumption totals for an interval measuring component that has a related scalar measuring component with the same UOM to ensure that the total consumption of the interval measuring component is within a tolerance of that of the scalar value. It can also be used to evaluate consumption totals for scalar TOU meters that have a "check" register (for example, three registers that measure ON-PEAK, OFF-PEAK, and SHOULDER, with a fourth check register that measures the total consumption).

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SUM-CHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-SumCheck

Unit of Measure Check

This rule checks the unit of measure (UOM) passed in with the Initial Measurement against the primary unit of measure configured on the measuring component's type.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-UOMCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-UOMCheck

Zero Consumption Check

This rule checks if the total consumption for the IMD is zero. There is also a check for whether an outage occurred during the same time as the zero consumption to provide ways to avoid To Dos by having a different exception type.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-ZEROCNCHK Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ZeroConsumptionCheck

Estimation VEE Rules

Interval Adjustment From Scalar

This rule adjusts interval Initial Measurement Data based on scalar values using one of two options configured on the VEE Rule. Both options require that a scalar measuring component be related to the current interval measuring component using the Consumption Reference Measuring Component, and that one or more final measurements be present for the related measuring component between the start and end date/times of the current Initial Measurement Data set.

1. Adjust all intervals. In this case, the scalar consumption provides a value that is then used to proportionally adjust all of the intervals in the set. The formula used is (Scalar Consumption / Total Initial Measurement Consumption) * Interval Amount. If the total of all of the intervals had been equal to zero originally, the rule adjusts all of the intervals to the same value.
2. Adjust intervals based on condition. For this option, a range of conditions is configured on the VEE Rule, and the rule adjusts only those intervals with conditions that lie within the configured range. The adjustment is done by applying an adjustment factor to each of the intervals within range.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTADJSCA Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalAdjustmentFrmScalar

Interval Averaging Estimation

The algorithm estimates when the current Measuring Component is eligible. Estimation is performed only if all these conditions are satisfied:

- The Measuring Component is interval.
- The Measuring Component is linked to a Service Point or the VEE Rule's Estimate If Not Attached to SP field is set to "Estimate”.
- The Percentage of Missing Intervals is less than the VEE Rule's Maximum Percentage Missing Intervals Threshold.

The algorithm will compute the Total Accumulated Consumption and the Total Number of Intervals using a variety of methods including scanning holidays, similar days of the week, or neighboring days. Once a set of valid comparison days are found then these are used as the basis of creating estimated intervals for the current IMD.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTAVGEST Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalAveragingEstimation

Interval Create Estimation IMD for Gap

This rule is very different from other VEE rules in that it does not examine the current initial measurement but rather looks to see if there are any missing measurements prior to the initial measurement being processed (i.e. a gap exists). If so and the scenario meets the configured options it will generate an estimation initial measurement to fill in that gap.

This rule is intended to provide more real-time filling of missing measurements as opposed to running periodic estimation. However, it is still expected that periodic estimation will be used in conjunction with this rule such that any gaps that are not filled in by this rule would eventually be filled in by periodic estimation.

This rule can be configured to perform minimal validation of that gap that is identified and defer to the estimation initial measurement to validate against other initial measurements and final measurements that may overlap the gap. Conversely, it can also be configured to validate the gap and exclude any periods where a final measurement or in progress initial measurement overlaps the gap's duration.

NOTE:
This estimation rule acts differently from the other rules - it creates a new IMD rather than filling in missing values in an estimation IMD created from periodic estimations.

Additional detail on the logic of this rule can be found in the Detailed Description of the D2-CREESTIMD Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

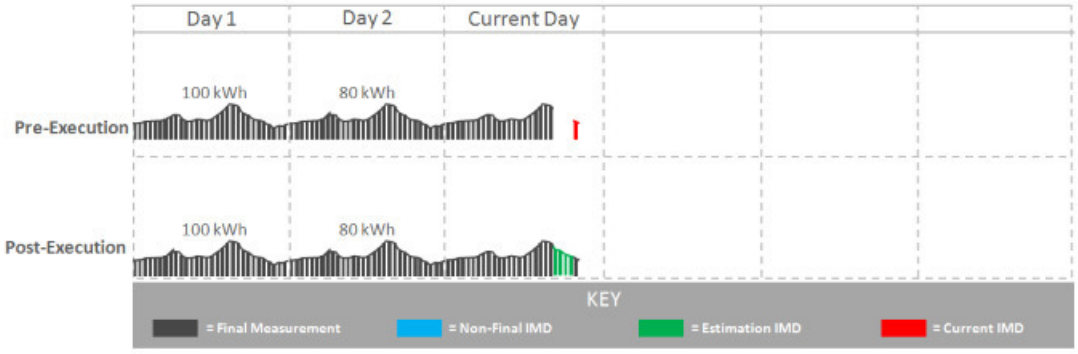
Business Object

D2-CreateEstimationIMDRule

Example Scenarios

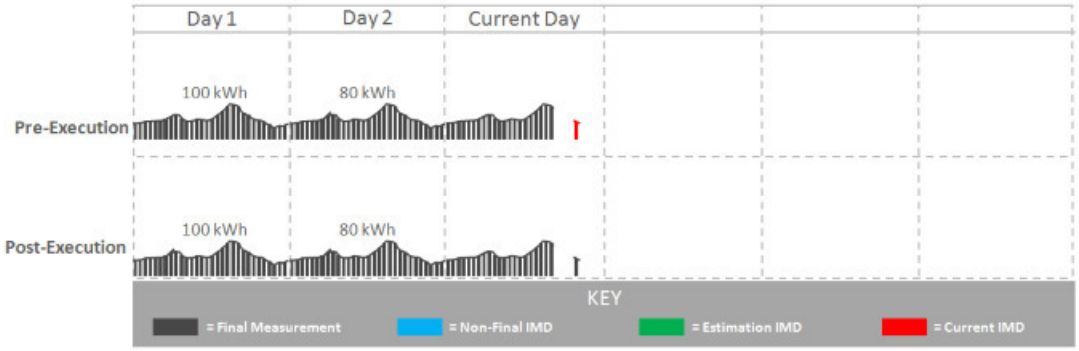
The following are sample scenarios that can be achieved based on configuration of this rule.

Example 1: A 4-hour gap exists and the rule is configured to always fill gaps. Maximum hours are set to 4 and the gap fill is set to "always".



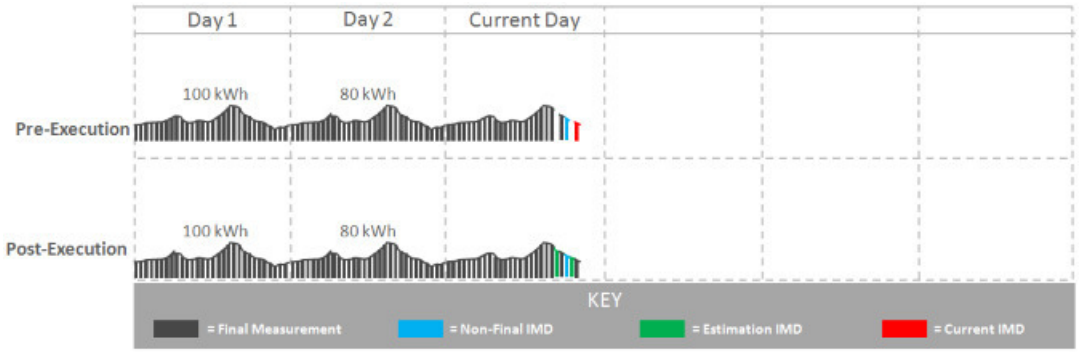
Since the maximum hours has not been exceeded and the rule is configured to always fill the gap, a 4-hour estimation IMD is created to fill the gap.

Example 2: A 4-hour gap exists and the rule is configured to always fill gaps. However, the maximum number of hours to fill is 3.



Since the gap is 4 hours long and the maximum hours to gap fill is 3 hours, no estimation has been created. An exception would be created if the rule were configured this way.

Example 3: A non-contiguous gap exists and the rule is configured to "Skip In-Progress IMDs and Final Measurements" with a maximum number of IMDs set to 2.



In this scenario:

- One interval in the 4-hour gap is covered by a final measurement.

- One interval in the 4-hour gap is covered by an in-progress IMD.
- Two estimation IMDs are generated one to fill each hour not covered by a final measurement or in-progress IMD.

Interval Interpolation Estimation

This rule attempts to interpolate gaps in Initial Measurement Data sets using prior and subsequent intervals as starting points for linear interpolation. A configuration parameter determines the maximum number of consecutive intervals within a gap before that gap can no longer be interpolated.

If a given gap lies at the beginning or end of the Initial Measurement Data set, this algorithm seeks out final measurements immediately before or after the set in an attempt to find two reference measurement values for interpolation. In this situation, assuming that the condition of the measurement retrieved lies in NEITHER of the algorithm parameter ranges you have configured (the measurement is neither "missing" nor an "outage"), this measurement value is then used as the basis for interpolation.

In the event that the gap is the entire length of the Initial Measurement Data set (and the VEE Rule is configured such that this is not too large of a gap), the routine attempts to find final measurements as described above.

If a valid measurement can be found for only one side of a given gap, the interpolation logic assigns each interval in the gap the same value - the value of the measurement retrieved. This is referred to as applying a "flat load".

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTINTEST Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalInterpolationEst

Interval Profile Estimation

This rule uses a profile measuring component's interval consumption as a source of values to assign to intervals in the current Initial Measurement Data set that are marked "missing" via the interval's condition. If the interval is marked with a condition indicating "outage", the algorithm skips it.

If a measurement is not available for the profile measuring component on the date/time, the interval is left unchanged.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTPROEST Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalProfileEstimation

Interval Proxy Day Estimation

This rule checks for days with similar temperature and uses the measurement data from those days as the basis of interval estimation for the in-flight initial measurement. Measuring component comparison periods are used as the main driver for determining similar temperature days.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-INTPROXY Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalProxyDayEstimation

Scalar Calculation From Interval

This rule calculates a single consumption amount for a scalar measuring component's Initial Measurement using the total consumption for the same date/time range for a related interval measuring component.

The scalar value replaces any existing value within the Initial Measurement (in the post-VEE list) and updates the condition to the value configured on the VEE Rule. This VEE Rule updates the condition to the value defined in "Condition Value for High Quality Condition" when the measurement condition value for all underlying interval data and the previous scalar measurement (when Evaluate Condition Of Previous Scalar Measurement is configured as "Yes") is greater than or equal to the "Minimum Condition Quality to Override" (If "Minimum Condition Quality to Override" is defined) otherwise it updates the measurement condition of the Initial Measurement Data to the condition defined in "IMD Created Condition Value".

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SCACALINT Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ScalarCalcFromInterval

Scalar Estimation

This rule uses historical data for the same measuring component to derive an estimated value for a scalar Initial Measurement. Depending upon a VEE Rule configuration parameter, the routine uses historical data from a month ago followed by a year ago, or vice-versa. If the data for the first historical period turns out to be usable, the second historical period will not be evaluated. Whether historical data qualifies for use in estimation is configured via the rule parameters in conjunction with the below algorithm parameters.

The rule rejects consumption from a historical period as unusable for estimation if too great a portion of the period is covered by final measurements that are not high-quality. The first pair of parameter values are used when evaluating the acceptable System-Estimated Percentage for historical periods as configured on the VEE Rule. Once an estimated value

is calculated, the routine backs into a reading (which involves backing out multipliers and, if the measuring component is subtractive, adding the result to the prior reading).

An interim High/Low can also be configured on this rule.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SCALAREST Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ScalarEstimation

Scalar Profile Estimation

This VEE rule arrives at a scalar estimate by looking at final measurements for a profile measuring component covering the same date range as the current Initial Measurement. The profile measuring component to be used as a source of measurement data is defined via a profile factor on the rule. This rule is meant primarily for a configuration in which the profile measuring components are interval, although the profile could be scalar as well.

If a measurement already exists in the current Initial Measurement with a condition that lies within the ranges specified in the pairs of algorithm parameters for "system estimate" and "regular", the routine does not attempt to estimate.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SCAPROEST Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ScalarProfileEstimation

Scalar Proration

This rule prorates the value of a scalar reading that has two valid scalar readings on either side as boundaries.

If the scalar MC does have a related interval MC and 'Minimum Related Measurement Condition' is defined the algorithm will take into consideration any intervals during the proration period that meet a minimum condition quality defined by the 'Minimum Related Measurement Condition' when performing the proration. This is to keep the prorated scalar estimates in sync with any existing interval measurements. To do this the algorithm will augment the total duration and total quantity being prorated by the related interval MC's qualifying measurements. For example, take the following scenario:

- Scalar proration is executing for a scalar IMD for 01/01 12:00 AM - 01/02 12:00 AM (24 hours)
- The subsequent measurement is for 150 kWh and has a measurement date/time of 01/04 12:00 AM (72 hours)
- The related interval MC has measurements on 01/03 12:00 AM to 01/04 12:00 AM totally 100 kWh (24 hours)

Therefore the result of the calculation for the scalar IMD from 01/01 12:00 AM to 01/02 12:00 AM would be 25 kWh. Since the interval measurements covered 24 of the 72 total hours of the 150 kWh that would leave 50 kWh to be split among the remaining 48 hours. Which leaves 25 kWh for 01/01 12:00 AM to 01/02 12:00 AM and 25 kWh for 01/02 12:00 AM to 01/03 12:00 AM.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SCLRPRORT Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ScalarProration

Subtractive Interval Adjustment Rule

This rule adjusts qualifying consecutive intervals for a subtractive interval initial measurement (IMD) based on the known consumption between a set of actual readings. The readings can either be leveraged from other intervals within the IMD or from the final measurements when there are no suitable readings within the IMD. The adjustment is done by applying an adjustment factor to each of the qualifying intervals.

The rule supports adjusting multiple sets of qualifying consecutive intervals by calculating a consumption adjustment target and adjustment factor specific to each set of consecutive qualifying intervals.

Intervals are considered to be qualified for adjustment based on condition. A range of conditions is configured on the VEE Rule, and the rule adjusts only those intervals with conditions that lie within the configured range.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SUBINADJV Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-SubIntAdjustmentVEE

Decision-Making VEE Rules

Exception Handler

This rule allows you to terminate the VEE process based on a configurable set of exceptions being present for the IMD. This rule also allows a unique To Do Type to be generated based on a group of exceptions.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-AVEER-EEH Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D1-VEERuleExceptionHandler

Execute VEE Group

Many implementations need to execute a group of validations for any data being received. For example, you might want to perform device identifier validations, multiplier checks, and UOM checks on all measuring components. One inefficient way to meet this requirement would be to repeat these three rules in multiple VEE groups. However, this solution becomes hard to maintain if changes to the rules are required (or if new "global rules" are introduced) as each group would have to be updated. Instead of this, you can use the "Execute VEE Group" rule to execute a referenced VEE Group. Using the example above, you could create a group called "Rules for All MCs" that contains a device identifier validations rule, a multiplier check rule, and a UOM check rule, and then reference the "Rules for All MCs" group in a Execute VEE Group rule.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-AVEER-RFG Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D1-VEERuleReferredVEEGroup

Successful Termination

This rule allows VEE to be successfully terminated based on a list of exceptions.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-AVEER-EST Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D1-VEERuleSuccessTermination

VEE Group Matrix (Factor)

Another situation likely to occur in many implementations is where specific rules may need to be applied to measurement data based on specific criteria, such as geography. For example, some geographic territories may have unique VEE Rules in addition to rules that are applied to all geographic territories. The "VEE Group Matrix (Factor)" rule allows for a Factor to determine which VEE Rule gets executed based on defined characteristics.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-AVEER-FCT Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D1-VEERuleGroupFactor

Prerequisite Setup

Since the prerequisite setup items for this rule are more involved, the following procedure describes these in detail:

1. Create the Characteristic Type and Values to be used by the factor that will be referenced by the rule.
2. Create the Characteristic Source Algorithm to be used by the factor that will be referenced by the rule.
3. Create the VEE Groups to be associated to the characteristic values.
4. Create the Factor that will be referenced by the rule.
5. Create the Factor Values for the factor, each referencing an effective-dated characteristic value/VEE group pairings.
6. Create the rule, referencing the factor

Measuring Component Statistics

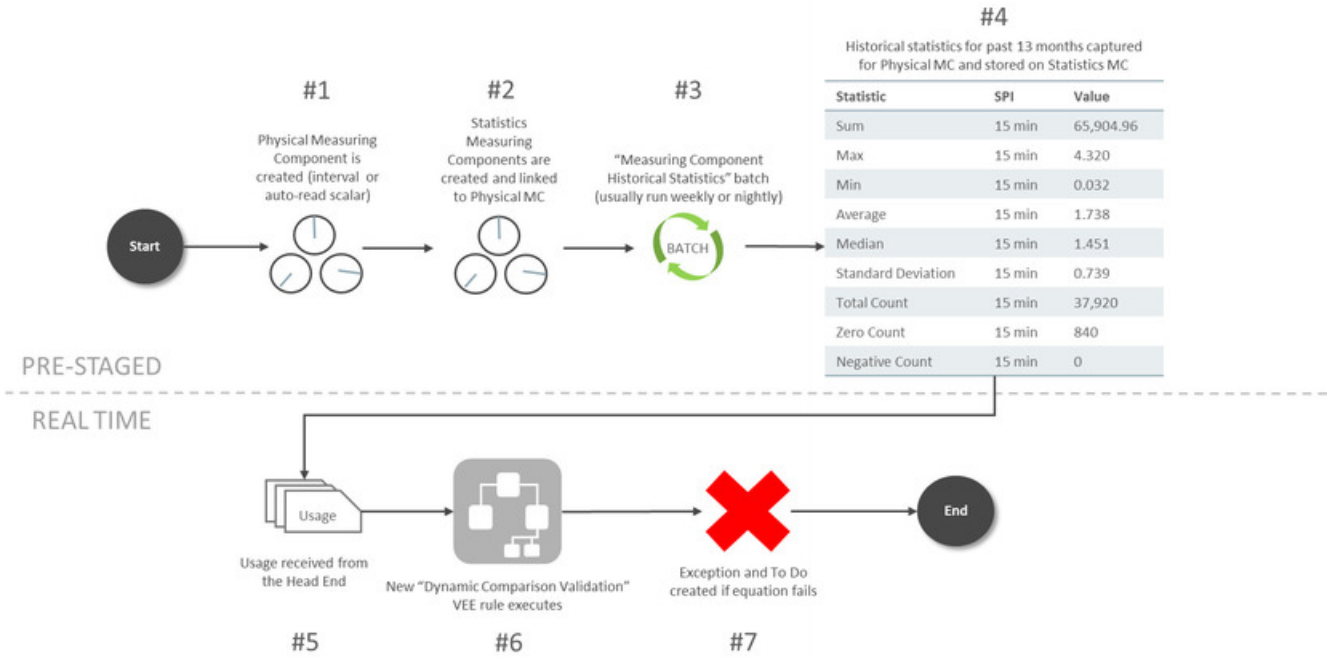
Understanding Measuring Component Statistics

Oracle Utilities Meter Data Management provides a way to stage statistics for Measuring Components in order to speed up VEE processing. These statistics are primarily meant for use in conjunction with the Dynamic Comparison Validation VEE Rule, but could also be generated for other analytics or reporting.

Some of the potential benefits of using Measuring Component Statistics are:

- Reduce the data volume for VEE queries by up to 1000 times (this is most beneficial when replacing VEE Rules that query historical data such as High/Low)
- Offload heavier processing of historical data to "system downtimes" - potentially nights or weekends

Below is an overview example of how Measuring Component Statistics can be implemented along with the Dynamic Comparison Validation:



The steps for pre-staging Measuring Component Statistics are detailed below:

1. A Physical Measuring Component is created. The algorithm to trigger the creation of Statistics Measuring Components is only set on Interval or Auto-Read Scalar BOs as part of base product.
2. New Statistics Measuring Component are created and linked to the Physical Measuring Component.
3. The Measuring Component Historical Statistics Batch (D1-MCHS) is run based on the schedule you've defined. The suggestion is to run this either nightly or on the weekends during quieter times for overall system processing.
4. A measurement is created for the Statistics Measuring Component that holds historical statistics related to the Physical Meter for a specific period of time (in this example it's set for 13 months).

The steps for how VEE can take advantage of Measuring Component Statistics are detailed below:

1. IMDs are received from the Head End.
2. The Dynamic Comparison Validation VEE Rule is configured to execute for the meter. This validation uses the historical statistics from the Statistics Measuring Component in order to perform its validation logic.
3. An exception and potentially a To Do is generated based on the failure of the VEE rule.

Storage Impact from Statistics

Much consideration was given to the storage implications when designing Measuring Component Statistics. Below is an outline of the storage impact to customers who choose to calculate and store Channel Statistics. An example impact to a utility with 1 million Physical Meters (each with 4 channels of information) is shown in the far right column:

Area	Additional Data in MDM	Example Impact (assuming utility has 1 million Physical Meters)
Additional MCs *Assumption: Statistics are only configured on the primary billing channel MC Type for the majority of your Physical Meters.*	1-3 extra MCs per Physical Meter *Assumption: customers will only configure Statistics on their primary billing channel MC Type for the majority of their Physical Meters.*	1 - 3 million extra MCs *This increases the Measuring Component table by an extra 25% to 75% from its size without statistics.*
Additional Measurements *If 'Statistics Retention Mode' configured as "Only Retain Most Recent Record".*	1 extra Measurement per Physical Meter	1 - 3 million extra Measurements for the life of MDM *This data is a minuscule fraction compared to the overall AMI data footprint.*
Additional Measurements *If 'Statistics Retention Mode' configured as "Retain Historical Records".*	1 extra Measurement per Physical Meter per week	1 - 3 million extra Measurements per week (52-156 million per year) *This data is 0.2-0.5% of the size of the AMI data collected in that same week.*

Configuring Measuring Component Statistics

Measuring Component Types

In order to start generating Measuring Component Statistics, two different configurations must occur for Measuring Component Types:

- New Measuring Component Types must be configured that leverage the **Channel Statistics Type** business object. These should be setup as the periods for which Measuring Component Statistics should be calculated. For example, if you

would like to be able to reference both last month, last year, as well as the past 13 months as statistics periods, then 3 different Channel Statistics Types must be setup for this. These could be setup as follows:

- • "Last Month" Channel Statistics Type:
  - Period End Lag Days = 0
  - Period Start Lag Days = 30
  - "Same Month Last Year" Channel Statistics Type:
  - Period End Lag Days = 365
  - Period Start Lag Days = 395
  - "Prior 13 Months" Channel Statistics Type:
  - Period End Lag Days = 0
  - Period Start Lag Days = 395

NOTE: *it's recommended to set the Statistics Retention Mode to "Only Retain Most Recent Record" on the Channel Statistics Type to reduce the amount of data being stored in Oracle Utilities Meter Data Management.*

- For a Physical Measuring Component Type leveraging either "Interval Channel Type" or "Register - Auto-Read Type", the **Related Statistics Measuring Component Types** section must be filled in with the appropriate types that should be created for the Physical Measuring Component.

VEE Rule

If you have a desire to leverage Measuring Component Statistics for VEE, the Dynamic Comparison Validation VEE Rule section should be referenced. A number of detailed examples of how to implement this rule are provided in this section as well.

Related Batch Controls

There are a few batches directly involved with Measuring Component Statistics:

- **Measuring Component Historical Statistics** (D1-MCHS): this batch monitors any active Statistics Measuring Components to execute the logic for calculating a new set of historical statistics.
- **Statistics Measuring Component Creation** (D1-STMCC): this batch is used to create Statistics Measuring Components on any Physical Measuring Components that were created prior to configuring this module. This is especially useful for existing customers that want to implement Statistics Measuring Components for all of their devices.

NOTE: Refer to Understanding Measuring Component Statistics for more information.

Chapter 13

Defining Usage Options

Usage Subscription Types

Understanding Usage Subscription Types

Usage Subscription Types define a collection of properties defining a class of Usage Subscriptions. Usage Subscription types also control valid values for various attributes of Usage Subscriptions.

For a deeper functional understanding, refer to the About Usage Subscriptions or About Usage Calculation sections of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Usage Subscription Types

This portal is used to display and maintain a Usage Subscription Type.

Refer to Understanding Usage Subscription Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Usage Subscription Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Usage Subscription Type List**: displays all of the Usage Subscription Types so you can choose the one you want to display in more detail
- **Usage Subscription Type**: shows the specific configuration for the selected Usage Subscription Type

The Usage Subscription Type defines a number of lists for "valid" objects that can be used in conjunction with the overall usage calculation process:

- Valid Usage Recipients
- Valid Service Point Types

- Valid Usage Calculation Groups
- Valid Dynamic Option Types

Integrating Usage Transactions

Requesting Usage Transactions from External Systems

When Oracle Utilities Meter Data Management is deployed with Oracle Utilities Customer Care and Billing usage transactions can be created via a request from Oracle Utilities Customer Care and Billing.

To invoke a usage transaction request, Oracle Utilities Customer Care and Billing invokes a Usage Transaction Request Inbound service script which invokes the Usage Transaction Seeder (D2-UsgTranSeeder) business object. This business object does the following:

- Determines the usage subscription ID based on an external usage subscription ID
- Determines the appropriate usage transaction business object to create. This uses the "How To Create Usage Subscription Related Information" processing method defined for the "Usage Transaction Creation" processing role on the service provider that represents the external system.

For a deeper functional understanding of Usage Transactions, refer to the About Usage Transactions section of the *Oracle Utilities Meter Solution Business User Guide*.

Processing and Sending Usage Transactions to External Systems

When Oracle Utilities Meter Data Management is integrated with a customer information system (such as Oracle Utilities Customer Care and Billing), usage transaction information, including bill determinants and other data can be sent to the customer information system (again, such as Oracle Utilities Customer Care and Billing).

Usage transactions are sent to subscribing systems when they enter the "Sent" state. The "Send Usage" (D2-SEND-USG) Enter algorithm on the Sent state of the base package usage transaction business object (D2-UsageTransaction) determines the method used to send the information, based on the Usage Recipient (service provider) defined on the usage transaction's Usage Subscription. Usage transactions can be sent to service providers via either online real-time processing or periodically via batch processing.

Online Real-Time Processing

To set up the service provider to support online real-time notification of usage transactions, do the following:

- Create one or more Outbound Message Types that reference the outbound message business object to be used to send usage transaction information to the external system. The base package include the following business object for Usage Transaction Outbound Message (D2-UsageTranOutboundMesg).
- Define a Message Sender that will be used to send the message to the external system.
- Add the outbound message type to the service provider's External System and reference the Message sender created above.
- Add a processing method to the service provider as follows:
  - **Processing Role**: Usage Transaction Notification - Online
  - **Processing Method**: How To Send US Related Information

- **Status**: Active
- **Default Processing Method**:
- **Outbound Message Type**: the outbound message type created above
- **Override Processing Method**: outbound message types for specific usage subscription types if applicable

For a deeper understanding of integrating usage transactions with a customer information or Oracle Utilities Customer Care and Billing, refer to the Configuring the Bill Determinant Interface in the Interface section.

Batch Processing

If Oracle Utilities Meter Data Management is integrated with a customer information system (such as Oracle Utilities Customer Care and Billing) that requests bill determinants, usage transaction processing should be coordinated with the requesting system's billing process.

Requests from a billing system that have been indicated as "batch requests" (such as those produced by Oracle Utilities Customer Care and Billing batch billing process) accumulate in a "calculation deferred" state to be processed specially by the **Usage Transaction Calculate Defer Monitor** batch control (D2-UTCD).

To set up the service provider to support periodic batch processing of usage transactions, use a periodic monitor batch control. These batch programs should invoke the business objects that will contain the usage transaction information. The base package includes the following business object for this: Usage Transaction Outbound Message (D2-UsageTranOutboundMesg).

Next, setup the following configuration:

- Add a processing method to the service provider as follows:
  - **Processing Role**: Usage Transaction Notification - Batch
  - **Processing Method**: How To Send US Related Information
  - **Status**: Active
  - **Default Processing Method**:
  - **Batch Control**: the batch control created above
  - **Override Processing Method**: batch controls for specific usage subscription types if applicable

Errors encountered when processing Usage Transactions can be reprocessed with the following batches:

- Reprocessing usage transaction "seeders" in error (D2-UTSED)
- Reprocessing usage transactions in error (D2-UTID)

If unexpected errors occur that leave usage transactions in an unmonitored state, the **Usage Transaction Monitor** batch control (D2-UT), or one based on this batch control with parameter values tailored to any specific requirements, can be used to process those usage transactions.

For a deeper functional understanding of Usage Transactions, refer to the About Usage Transactions section of the *Oracle Utilities Meter Solution Business User Guide*.

Generating Usage Transactions in Oracle Utilities Meter Data Management

There may be some cases where Usage Transactions should be generated in Oracle Utilities Meter Data Management rather than requested from an external system. Generation of these usage transactions are scheduled around the time when the meter installed on the service point is read.

Base product delivered a batch program that reads all pending measurement cycle schedule routes with selection date on or before the business date and creates as SP measurement cycle schedule route business object instance that stages the creation of these usage transactions.

For additional details refer to the D1-CSPSR batch control and D1-SPMsrmtCycScheduleRoute business object.

Usage Transaction Exception Types

Understanding Usage Transaction Exception Types

Usage Transaction Exception Types define the groupings of exceptions for a Usage Transaction based on their functional similarity. This provides a way to define Usage Transaction Exceptions in a distinct enough way to understand the root issue that was generated from the usage calculation rule.

For a deeper functional understanding of Usage Calculation, refer to the About Usage Calculation section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Usage Transaction Exception Types

This portal is used to display and maintain a Usage Transaction Exception Type.

Refer to Understanding Usage Transaction Exception Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Usage Transaction Exception Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Usage Transaction Exception Type List**: displays all of the Usage Transaction Exception Types so you can choose the one you want to display in more detail
- **Usage Transaction Exception Type**: shows the specific configuration for the selected Usage Transaction Exception Type

Usage Calculation Groups

Understanding Usage Calculation Groups

Usage calculation groups are collections of usage calculation rules that are used to either calculate bill determinants or validate bill determinants. During the usage transaction process, the system executes the usage calculation rules defined in the usage calculation group referenced on the usage subscription. The rules within a usage calculation group are defined in a specific sequence, allowing control over the order in which the rules are executed.

Usage calculation groups are associated with specific usage subscriptions and usage subscriptions types (or both). When assigned to usage subscriptions, usage calculation groups contain the usage calculation rules to be used to calculate usage / bill determinants. Usage calculation groups associated with usage subscription types are those groups considered valid for usage subscriptions of that type.

Usage calculation groups can also specify a list of device donfiguration types that are considered valid. Usage calculation groups should only be associated with usage subscriptions for service points related to device configurations of a valid device configuration type.

Usage calculation groups can also be referenced by the Execute Usage Calculation Group usage calculation rule.

For a deeper functional understanding of usage calculation, refer to the About Usage Calculation section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Usage Calculation Groups

This portal is used to display and maintain a usage calculation group.

Refer to Understanding Usage Calculation Groups for more information.

You can access the portal by selecting**Admin**, then **Usage**, then **Usage Calculation Group**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Usage Calculation Group**: Defines basic information about usage calculation group
- **Usage Calculation Rules List**: lists the usage calculation rules belonging to the group
- **Referencing Usage Calculation Rules List**: lists the usage calculation rules that reference the group
- **Referencing Usage Subscription Type List**: lists the usage subscription types that reference the group
- **Referencing Usage Subscriptions List**: lists the usage subscriptions that reference the group

Usage Calculation Rules

Understanding Usage Calculation Rules

Usage calculation rules are standard and custom rules that perform calculations on measurement data to generate bill determinants and other values used by external systems, such as billing systems, customer information systems, etc. Usage calculation rules are created for a specific usage calculation group. For example, if you were configuring two usage calculation groups and both included a specific usage calculation rule, you would need to create two instances of the usage calculation rule, one for each group.

The base package includes rules that calculate common bill determinants including:

- Scalar reads
- Time-of-use consumption (by applying a time-of-use map to an interval channel)
- Interval curves (either real or derived)
- Virtually anything else that can be calculated from the information in the system

For a deeper functional understanding of usage calculation, refer to the About Usage Calculation section of the *Oracle Utilities Meter Solution Business User Guide*.

Usage Transaction Exceptions

On almost every usage calculation rule, the failure of the rule results in a usage transaction exception and the Usage Transaction Exception Type for the failure can be configured on the rule. These usage transaction exception types can also be set to a specific Exception Severity:

- **Information**: Used to highlight minor issues, but not sufficient to cause the usage transaction to be put into a failure state. Exceptions of this category can be used to report on the frequency of interesting, but not fatal issues

- **Issues**: Used to report a problem that will prevent the usage transaction from being sent. Multiple "issue exceptions" can be created during usage transaction processing. If at least one issue exists after all rules have been applied, the usage transaction is transitioned to a failure state requiring review and approval.
- **Terminate**: Used to report a severe issue that will cause the usage calculation process to stop and the usage transaction to be transitioned immediately to a failure state requiring review and approval. Only one terminate exception can be issued (as the first one causes calculation processing to stop on for a Usage Transaction). This should be used for cases where manual override / approval isn't accurate. For example, a "Curve Not Continuous" error that says the interval data doesn't cover the full usage period should always be set to Terminate as an Exception Severity.

Rounding Usage Transaction Service Quantities

Several usage calculation rules provide a means for defining the manner in which service quantities are rounded during usage calculations. Service quantity rounding can be defined using the **SQ Rounding Details** which allow configuration of the method by which quantities are rounded (Up, Down, or Nearest) and the number of decimal positions retained after rounding.

Creating Measurement Data Snapshots

Daily Scalar Calculation Rule, Get Interval, Get Item Counts and Consumption, and Get Scalar Details usage calculation rules can be configured to create measurement data snapshots during usage transaction processing. Measurement data snapshots are used by data aggregation in Market Settlements Management.

Measurement data snapshot configuration is done in the **Extract To Measurement Data Snapshots** section, and includes the following options:

- **Measurement Data Snapshot Type**: The measurement data snapshot type for measurement data snapshots created by the usage rule.
- **Usage Transaction ID To Use**: A flag that designates the type of ID to use to identify the usage transaction on the measurement data extract. Options include:
  - **Usage Transaction ID**: The usage transaction’s ID
  - **Usage Transaction External ID**: The external ID for the usage transaction. Used when the usage transaction is triggered by external system such as a CIS (or by the billing functionality in Customer To Meter)
- **Extract To Different Usage Subscription**: A flag that designates whether or not the measurement data snapshot should be extracted to a different usage subscription, such as one used by Market Settlements Management. Options include:
  - **No**: The extract will reference the usage subscription for which usage calculations are being processed. This option is used when Meter Data Management and Market Settlements Management are installed in different application environments. In this case, the extract can be exported from Meter Data Management for subsequence import into Market Settlements Management.
  - **Yes**: The extract will reference a different usage subscription such as one used by Market Settlements Management. This option is used when Meter Data Management and Market Settlements Management are installed in the same application environment. In this case, the measurement data snapshot is “exported” within the system to the Settlement Item, where it can be used in data aggregation processing.
    When this option is selected, the **Source Identifier Type** and **Target Identifier Type** fields must be configured to define how the system will determine the correct settlement usage subscription to use.
- **Irregular Data Flag Mapping**: Ranges of condition codes used to flag data as “irregular”. The value of the **Irregular Data Flag** (based on the IRREGULAR_DATA_FLG lookup) is used to flag data that falls with each range configured. A blank **Irregular Flag** value denotes that the condition range is considered to be "Regular".

See About Measurement Data Snapshots for more information about measurement data snapshot creation and processing.

Configuring Usage Calculation Rules

This portal is used to display and maintain a usage calculation rule.

Refer to Understanding Usage Calculation Rules, Understanding Usage Calculation Groups, and Understanding Usage Transaction Exception Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Usage Calculation Rule**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Usage Calculation Rule**: Defines the usage calculation rule, including parameters used when executing the rule
- **Eligibility Criteria List**: Lists the eligibility criteria defined for the rule

Pre-Calculation Usage Calculation Rules

The following is a list of the pre-calculation usage calculation rules provided as part of base product. For more information on how each rule executes and can be configured, follow the link provided on the rule.

Usage Calculation Rule Name	Applicable Data Type(s)	Purpose
Alignment and Delay Usage Calculation Rule	Interval or Scalar	This rule can be used to handle two main needs: 1) aligns all measuring components for a Usage Subscription to the same date (whether on the same device or separate devices) 2) delays the usage transaction until the end of the retry window based on the quality of available data and other configured parameters.
Check Existence of Installed Device	Interval or Scalar	This rule checks for the existence of a device installed on the Usage Subscription's Service Point for the usage period. In the case of multi-items this rule also checks that they are effective for the usage period.

Calculation Usage Calculation Rules

The following is a list of the calculation usage calculation rules provided as part of base product. For more information on how each rule executes and can be configured, follow the link provided on the rule.

Usage Calculation Rule Name	Applicable Data Type(s)	Purpose
Apply Math (Interval Data)	Interval	This rule is used to perform calculations on interval data and stores the results in the usage transaction's service quantities. A variety of options are available on this rule that include defining the calculation type, variables to use, as well as the equation to use (math functions and expressions).

		This rule provides aggregated usage for all selected interval measuring components (filter by TOU, SQI & UOM) associated to a usage subscription. This rule can also multiply total usage by a factor using a custom formula.
Daily Scalar Usage Calculation Rule	Scalar	This rule is used to calculate usage of daily scalar measuring components installed in the Service Points associated with a Usage Subscription for the specified usage period. It creates bill determinants by taking the difference between the beginning and ending reading for the bill period. This rule can also be used to provide register readings by measuring component. For consumption values, only the beginning and ending readings are exported This rule supports date breaks (the normal Get Scalar Data rule does not).
Get Interval Data	Interval	This rule is used to get interval quantities from interval measuring components installed in the Service Points linked to the Usage Subscription for the specified 'Interval' usage period. This rule retrieves the interval data for measuring components associated to a usage subscription by TOU, SQI and UOM. This rule converts the interval data to another interval length or unit of measure.
Get Item Counts and Consumption	Interval or Scalar	This rule finds item-based and multi-item-based Service Points linked to the Usage Subscription for the current usage transaction, summarizes the item counts by item type and Service Point, and calculates item-based consumption.
Get Scalar Details	Scalar	This rule is used to get usage from scalar measuring components installed in the Service Points linked to the Usage Subscription for the specified 'Scalar' usage period. This rule creates bill determinants by summing all scalar readings for the bill period. This rule can also be used to provide register readings by measuring component. All readings are exported by this rule. Note: This rule is used for traditional monthly read meters.
Get Subtractive Interval Details	Get Subtractive Interval Details	This rule is used to get interval quantities from subtractive interval measuring components installed on the service points linked to the

		usage subscription for the specified 'interval' usage period. It also identifies the start and stop readings for each usage period using subtractive interval readings.
Get TOU Mapped Usage	Interval	This rule is used to get time of use quantities from interval measuring components for devices installed at the Service Points linked to the Usage Subscription for the specified 'Interval' usage period.
Interval Tier Calculation	Interval	This rule calculates the difference between a source and reference vector. This rule loops through each tier that is configured and calculates the imbalance amount associated to that tier level. This rule breaks down that difference into one to many positive or negative tiers, and create a service quantity for each tier calculated.
Profile Accumulation	Interval	This rule is used to manipulate a customer's interval data by adding other vectors to it. Those other vectors are derived from a list of profile factors and corresponding characteristic values stored in a list on the usage transaction.
Round and Adjust Usage	Interval or Scalar	This rule copies identified source and target Service Quantities and inserts these as Service Quantities that are rounded and adjusted.
Vector and Service Quantity Math	Interval	This rule is designed to facilitate configuration of complex vector calculations. It is based on a series of underlying services with vectors configured as input to the calculations. Typical uses: Perform math using interval data, e.g., take the difference between two curves, find max values, find coincident peaks, multiply a curve by a value, apply TOU maps, etc. Define complex formulas using various interval curves, profile factor values or calculated service quantities (bill determinant values). Support math functions: sin, cos, square root, etc. Store derived curves in memory that can be used in subsequent calculations Please note, this rule is not as efficient as other rules.

Post-Calculation Usage Calculation Rules

Below is a list of the post-calculation usage calculation rules provided as part of base product. For more information on how each rule executes and can be configured, follow the link provided on the rule.

Usage Calculation Rule Name	Applicable Data Type(s)	Purpose
Usage High/Low Rule	Interval or Scalar	This rule compares the Service Quantities of the Usage Transaction to historical values. If the current value is too high or too low compared to historical data then an exception is thrown.
Validate Against Tolerance	Interval or Scalar	This rule is used to validate the calculated usage against a tolerance value. The tolerance value may either come from the specified value or tolerance factor defined in the usage calculation rule.
Business Flag Hold	Interval or Scalar	This rule can stop a usage transaction from proceeding when there have been business flags for the applicable service points. The hold can either be indefinite or set to expire a configurable amount of time prior to the calculation window ending.

Decision-Making Usage Calculation Rules

There are usage calculation rules delivered as part of the base product that help with decision-making when executing running the usage calculation process. For more information on how each rule executes and can be configured, follow the link provided on the rule.

Usage Calculation Rule Name	Purpose
Execute Usage Calculation Group	This rule performs a call to execute a separate usage calculation group which includes execution of all usage calculation rules within that group.
Exception Handler	This rule is used to terminate execution of the usage processor if exception count criteria specified in the rule is met.

Advanced Aside: Using Factors For Variables

A situation common in some implementations involves converting one unit of measure (UOM) to another. However, the conversion factor used in conversions of this can differ based on many different types of criteria, such as the location of the service point or other characteristics. This sort of calculation can be implemented as a usage calculation rule that accumulates consumption for one UOM and converts the consumption to a different UOM by applying a factor to it.

Factors used for this purpose have a Factor Class of "Number," and use some unique rules:

- Number factors reference a characteristic type (with pre-defined values).
- Number factors reference an algorithm that retrieves or derives the value of the characteristic type at runtime.

Factor values for a Number factor are effective-dated pairings of a characteristic value and a corresponding value. Because these pairings are effective-dated, the value returned from the factor can change over time for each characteristic value. At run time, the rule retrieves / derives the characteristic value for the factor's characteristic type and then finds the value

associated with the respective characteristic value. Factors can be related to any real or dynamic attribute, so rules of this type are very flexible. For example:

- **Real Attribute**: you could create a rule that retrieves a specific value based on the location of a service point.
- **Dynamic Attribute**: you could create a rule that retrieves a percentage value based on the amount the customer conserved as compared to the same period in the prior year, returning one value if the amount conserved is between 5% and 10%, another value if the amount conserved is between 10% and 20%, and yet a third value if the amount conserved is greater than 20%. The amount conserved is dynamically calculated at execution time and is compared to the characteristic values defined for the factor, and returns the appropriate value. In this example, if the amount conserved was anything less than 5%, no percentage value would be returned.

Pre-Calculation Usage Calculation Rules

Alignment and Delay Usage Calculation Rule

This rule attempts to delay usage calculation until high quality data for calculating bill determinants have become available. It is especially useful for usage subscriptions having multiple sources of usages such as multiple service points or multiple measuring components on the installed meter. The rule ensures that the usage calculated is based on a common date or alignment date, identified for these sources.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-ALGNDELAY Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-AlignmentandDelay

Check Existence of Installed Device

This rule checks for the existence of a device installed on the Usage Subscription's Service Point for the usage period. In the case of multi-items this rule also checks that they are effective for the usage period.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-CHKEXTDVC Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ChkExistenceofInstalledDvc

Calculation Usage Calculation Rules

Apply Math (Interval Data)

This rule is used to perform calculations on interval data and it stores the results in the usage transaction's service quantities. A variety of options are available on this rule that include defining the calculation type, variables to use, as well as the equation to use (math functions and expressions).

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-APPMATHIN Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ApplyMathInt

Example Scenarios

Below are some example scenarios that can be achieved based on configuration of this rule.

Scenario 1: Get the total KWH for the period

Calculation Type = Single Variable

Variable Name = V1, where the following is configured:

- Variable Type = Channel Accumulation
- UOM = KWH
- Set Function = Sum

Scenario 2: Get the higher value between the total KWH or total KVARH

Calculation Type = Math Function

Math Function = Max

Variable Name = V1, where the following is configured:

- Variable Type = Channel Accumulation
- UOM = KWH
- Set Function = Sum

Variable Name = V2, where the following is configured:

- Variable Type = Channel Accumulation
- UOM = KVARH
- Set Function = Sum

Scenario 3: Total KWH multiplied with a factor value

Calculation Type = Mathematical Expression

Mathematical Expression = V1 * V2

Variable Name = V1, where the following is configured:

- Variable Type = Channel Accumulation
- UOM = KWH
- Set Function = Sum

Variable Name = V2, where the following is configured:

- Variable Type = Factor

- Factor = factor code that holds the multiplier

Daily Scalar Usage Calculation Rule

This rule is used to calculate usage of daily scalar measuring components installed in the Service Points associated with a Usage Subscription for the specified 'Interval' usage period. Only automatically read scalar measuring components with value identifier that matches usage calculation rule's UOM/TOU/SQI will be processed.

If configuring this rule with interval and scalar usage calculation rules under a usage calculation group, it should be executed after interval usage calculation rule and before the scalar usage calculation rule.

This rule calculates usage based on the beginning and ending reading values -- ignoring intermediate readings. This rule will not estimate data based on the usage transaction estimate flag. Frequently read scalar devices must use periodic estimation processes.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-GETFRESCR Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-CalFrequentlyReadScalar

Get Interval Data

This rule is used to get interval quantities from interval measuring components installed in the service points linked to the usage subscription for the specified 'Interval' usage period. Only measuring components that match the UOM/SQI defined in the usage calculation rule are processed. Measurements within the period are stored in the usage transaction's service quantities' interval data list. The SQ entry's quantity is calculated based on the **Calculate Function** defined in the usage calculation rule. This is done for every entry in the usage period list. Each service quantity recorded by this rule is linked to a service point and measuring component.

By supplying the Axis Conversion parameters on the usage calculation rule, this algorithm will convert the identified measuring component's consumption into the supplied UOM and interval size prior to storing the consumption into the service quantity's interval data list.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-GETINTDAT Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-GetIntervalData

Get Item Counts and Consumption

This rule finds item-based and multi-item-based Service Points linked to the Usage Subscription for the current usage transaction, summarizes the item counts by item type and Service Point, and calculates item-based consumption. Once the item detail entries are created, the rule processes goes through them to create usage period SQs.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-GETITEMCC Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-GetItemCountsConsumption

Get Scalar Details

This rule is used to get usage from scalar measuring components installed in the Service Points linked to the Usage Subscription for the specified 'Scalar' usage period. By default all scalar measuring components are processed but if specific UOMs/TOUs/SQIs are defined in the usage calculation rule then only applicable measuring components are processed. Measurements within the period are retrieved. The usage transaction request may indicate whether or not 'Estimate' measurements are allowed.

The measurement details are stored in the usage transaction's scalar details. The usage is also stored in the usage transaction's service quantities unless otherwise specified in the usage calculation rule (using Build Service Quantity indicator).

This rule can be configured to perform measurement quality assessment, which will result in the population of the measurement quality list with the quantities and corresponding conditions of those measurements (regular, estimated, etc.).

Please note: this rule should not be used with readings occurring daily or more frequently as it will retrieve all of the readings to store in the Usage Transaction. The Daily Scalar usage calculation rule may be a better option for this use case.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-GETSCALAR Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-GetScalar

Get Subtractive Interval Details

This rule is used to calculate the usage for a subtractive interval measuring components. This differs from the Get Interval Data usage calculation rule in that it also retrieves a start and stop reading for each usage period it calculates. The start and stop readings are then used to provide entries into the usage scalar details table which are then made available in the usage transaction sent to the billing system for bill presentment.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-SUBINTADJ Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-GetSubtractiveInterval

Get TOU Mapped Usage

This rule is used to get time of use quantities from interval measuring components for devices installed at the Service Points linked to the Usage Subscription for the specified 'Interval' usage period. Only measuring components that match the UOM/SQI defined in the usage calculation rule instance are processed.

Measurements within the period are mapped to time of use quantities based on the TOU map defined on the usage calculation rule. If dynamic options are specified in the referenced TOU map and if there are dynamic option events in effect within the usage period then the TOU map associated with the dynamic option is used for the entire dynamic option event period. This calculation is performed for every usage period requested.

The calculated time of use quantities are stored in the usage transaction's service quantities per Service Point and measuring component.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-GETTOUUSG Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-GetTOUUsage

Interval Tier Calculation

This rule calculates the difference between a source and reference vector. It then breaks that difference into one to many positive or negative tiers. For each tier calculated it will create a service quantity with the optional ability to create additional service quantities for the total of all positive tiers and the total of all negative tiers. This rule will loop through each tier that is configured and calculate the imbalance amount associated to that tier level.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-GETINTIER Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-IntervalTierCalculation

Profile Accumulation

This rule is used to manipulate a customer's interval data by adding other vectors to it. Those other vectors are derived from a list of profile factors and corresponding characteristic values stored in a list on the usage transaction.

The typical application of this rule is for customer self-service rate compare, in which a self-service user has selected a set of usage adjustments to apply to his/her historical consumption to assess the effect on the amount consumed.

This rule algorithm retrieves the customer's usage (the source UOM) for the usage period using one of two options:

- A channel linked to a Usage Subscription
- A usage transaction service quantity

Then it takes each entry in the usage transaction's profile factor list, finds the profile measuring component for each, retrieves the measurement data for the profile, applies axis conversion to align it to the source UOM and target interval size, and adds it to the source vector. Each profile's data is added progressively to arrive at a final vector. The final vector may then be TOU-mapped, and the vector itself may be preserved or discarded.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-DYNPRFLAC Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ProfileAccumulation

Profile Service Quantity

This rule intervalizes scalar service quantity values based on a profile measuring component.

This rule first retrieves scalar service quantities, identified by the rule's UOM/TOU/SQI. If multiple service quantities exist for the service point they will be summarized at the service point level. In the case of a split or broken usage period for a service point, the usage period is merged and the quantities are summarized. Each service quantity is subject to intervalization. The profile measuring component is then derived using the profile factor supplied with either the usage subscription or service point as characteristic sources. If the profile is not available, an exception will be created using the exception details on the rule.

If a dynamic option type is provided, the events associated with it are retrieved for the usage transaction period. Any intervalized quantities that coincide with the dynamic option events start/end date time are then recorded in a new usage period service quantity of usage type “Correlated SQ”.

If no dynamic option type provided, the intervalized quantities are recorded in a new usage period service quantity of usage type “Profile SQ”.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-PRFSVCQTY Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ProfileServiceQuantity

Round and Adjust Usage

This rule copies identified source and target Service Quantities and inserts these as period Service Quantities that are rounded and adjusted. This rule captures the source and target Service Quantity identifiers, rounding method, and Service Quantity bucket to hold the adjustment. The usage calculation rule also has an option to validate the difference between the source and target service quantities.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-RNDADJUSG Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-RoundAndAdjustUsage

Vector and Service Quantity Math

This rule is designed to facilitate configuration of complex vector and scalar quantity calculations. It is based on a series of underlying services, including Axis Conversion, Apply Formula (most importantly), and Apply TOU Map. It can be configured to accept as input up to five vectors - where a single vector can represent the combination of all measuring components with linkages to the Usage Subscription with the configured UOM/SQI. It can also accept a list of scalar variables that can be taken in as factor values, usage transaction service quantities, set functions on a vector, or numeric values.

Vectors can be combined using a simple formula expression, or using condition formula expressions. Both interval values and interval conditions can be referenced. The result is interval-by-interval processing of the vector formula. The resulting final vector can be stored, TOU mapped, and/or subjected to a final set function (such as sum or max).

This rule stores service quantities of type "other" - meaning that the service quantities created by this rule do not have linkages to a specific Service Point or measuring component, given the potentially diverse sources of the data taken as input by this rule that lead to the final quantity.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-MATH Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-Math

Post-Calculation Usage Calculation Rules

Usage High/Low Rule

This rule is used to validate the current usage against historical usage - previous year usage or previous usage. It ensures that any increase or decrease of the current usage relative to historical usage is within a tolerance.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-USGHIGLOW Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-UsageHighLowRule

Validate Against Tolerance

This rule is used to validate the calculated usage against a tolerance value. The tolerance value may either come from the specified value or tolerance factor defined in the usage calculation rule.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-VALUSGTOL Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ValAgainstTol

Business Flag Hold

This rule can be used to place a hold on a usage transaction until the business flags related to the usage transactions service points can be investigated. The hold can be placed in one of two ways:

- **Calculation Window Based**: holds will remain until either a user manually bypasses the usage exception error or the current date is within a configurable tolerance of the calculation window end (aka the retry until date time).
- **Indefinite**: holds require a user to manually bypass the usage exception that is generated.

Defining which business flags should result in a usage transaction is a matter of identifying which business flag types, priorities, and confidences should be included.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-AUR-IBFH algorithm type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-BusinessFlagHold

Decision-Making Usage Calculation Rules

Execute Usage Calculation Group

This rule performs a call to execute a separate usage calculation group which includes execution of all usage calculation rules within that group. It is especially useful in cases where repeating the same rule over and over would be hard to maintain.

For example, you may want to calculate a straightforward kWh sum for every usage calculation group. Creating a separate CALC_KWH_SUM usage calculation group for this one calculation allows you to isolate the configuration points. Then the

CALC_KWH_SUM usage calculation group can be referenced in every other usage calculation group that needs this sort of service quantity.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-AUSGR-RFG Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D1-UsageRuleReferredUsageGroup

Exception Handler

This rule evaluates the exception list that was accumulated during the execution of the calculation rules against exception criteria configured on this rule. This used to terminate the execution of usage calculation group processor if there are far too many exceptions hit during the execution.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-UREXPCHAN Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-UsageRuleExceptionHandler

Detailed Configuration Examples

Demand Calculation Options Using Interval Data

There are a number of ways to calculate a demand value from interval data when retrieving bill determinants for CIS. A few examples of demand calculations are listed below:

Scenario1: Use interval data for demand at a common block size: the Vector and Service Quantity Math usage calculation rule can be used to calculate demand from the interval data. Below is one example of configuration to handle demand calculation based on 30 minute blocks:

Usage Calculation Rule field	Value	Notes
Vector 1 // Type	Physical Channels Links to Usage Subscription	
Vector 1 // Unit of Measure	Kilowatt-Hours	Use your own UOM here - this is just as example.
Vector 1 // Service Quantity Identifier	Consumed	Use your own SQI here - this is just as example.
Vector Processing // Common Interval Size	00:30:00	This configuration rolls up all channels (whether 5 minute, 15 minute, or 30 minute interval data) into a common interval size.

Vector Processing // Vector Formula Source	Simple Vector Formula	
Vector Processing // Simple Vector Formula	IV1*2	This is multiplying the common 30 minute interval size by 2 to achieve an hourly value.
Result // Unit of Measure	Kilowatts	Use your own UOM here - this is just as example.
Result // Service Quantity Identifier	Consumed	Use your own SQI here - this is just as example.
Result // Insert Primary SQ Entry	Yes	
Result // SQ Entry Quantity Source	Set Function Against Derived Vector	
Result // Set Function Against Derived Vector	Max	This function pulls the max value based on the configuration in the "Vector Processing" section.

Scenario 2: Use interval data for demand by TOU period: This example is very similar to the last and again leverages the Vector and Service Quantity Math usage calculation rule. However, instead of calculating a single demand value for the entire period it will calculate a demand value for each TOU period:

Usage Calculation Rule field	Value	Notes
Vector 1 // Type	Physical Channels Links to Usage Subscription	
Vector 1 // Unit of Measure	Kilowatt-Hours	Use your own UOM here - this is just as example.
Vector 1 // Service Quantity Identifier	Consumed	Use your own SQI here - this is just as example.
Vector Processing // Common Interval Size	00:30:00	This configuration rolls up all channels (whether 5 minute, 15 minute, or 30 minute interval data) into a common interval size.
Vector Processing // Vector Formula Source	Simple Vector Formula	
Vector Processing // Simple Vector Formula	IV1*2	This is multiplying the common 30 minute interval size by 2 to achieve an hourly value.
Result // Unit of Measure	Kilowatts	Use your own UOM here - this is just as example.
Result // Service Quantity Identifier	Consumed	Use your own SQI here - this is just as example.
Result // Insert Primary SQ Entry	No	
Result // Apply TOU Map To Derived Vector	Yes	
Result // TOU Map	(your TOU Map)	Select the TOU Map you'd like to apply.
Result // Time Of Use Calculate Function	Max	This function pulls the max value based on the configuration in the "Vector Processing" section and then buckets it based on the selected TOU Map.

Usage Calculation Types

Understanding Usage Calculation Types

Usage calculation types define calculations executed at a customer level where there may be more than one calculation per usage subscription. For example, a customer may have a usage subscription for producing billing determinants but there may be additional calculations that will be executed on a much less frequent basis (annually, quarterly, etc). Usage calculation types provide a way to leverage a single usage subscription to drive multiple different calculations.

Usage calculation types use the following parameters:

- **Request Type**: Used to drive the creation and processing of the usage transactions for all usage subscriptions that are valid for the usage calculation type.

  NOTE: Request types for usage calculation types should be based on the Usage Calculation Request Type business object (D2–UsageCalculationReqType)

- **Service Point Quantity Creation**: Details of the quantities created by usage calculations of this type, including service point quantity type, and UOM, TOU, and SQI.
- **Applicable Usage Subscription Types**: A list of usage subscription types for which this usage calculation type is applicable. Each usage subscription type is defined by an effective date range, the usage transaction business object used for the usage calculation, the result UOM/TOU/SQI, and the usage calculation group used to perform the calculations.

Configuring Usage Calculation Types

You use the **Usage Calculation Type** portal to maintain and view usage calculation types.

Refer to Understanding Usage Calculation Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Usage Calculation Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Usage Calculation Type List**: This zone lists all usage calculation type records. Broadcast a record to display the details of the selected record.
- **Usage Calculation Type**: This zone displays details for the selected usage calculation type.

Usage Subscription Quantity Types

Understanding Usage Subscription Quantity Types

Usage subscription quantity types define types of quantities that can be stored for a usage subscription. These quantity types are often infrequently calculated descriptors of a given customer and how they relate, at an aggregate level, to either the customer base as a whole or their particular rate class. For example, a customer quantity may be used to store a scaling factor that describes how a given customer’s usage compares to the profiled usage for their rate class.

In a settlement implementation, usage subscription quantity types can be used to define values calculated monthly, annually, etc., such as annual peak load contribution (PLC).

Usage subscription quantity types use the following parameters:

- **Service Type**: The service type (electric, gas, water, etc.) for service point quantities of this type.
- **Usage Subscription Quantity Business Object**: The business object used for usage subscription quantities of this type.

- **Usage Subscription Quantity Identifiers**: Value identifiers related to the current usage subscription quantity type (used to provide shorthand descriptions of the various types of values measured by usage subscription quantities of this type).

Configuring Usage Subscription Quantity Types

You use the **Usage Subscription Quantity Type** portal to display and maintain usage subscription quantity types.

Refer to Understanding Usage Subscription Quantity Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Usage Subscription Quantity Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Usage Subscription Quantity Type List**: This zone lists all usage subscription quantity type records. Broadcast a record to display the details of the selected record.
- **Usage Subscription Quantity Type**: This zone displays details for the selected usage subscription quantity type.

Dynamic Option Types

Understanding Dynamic Option Types

Dynamic options specify terms that override how usage is normally calculated, such as a critical peak period that affects the TOU mapping of interval consumption. Dynamic option types store information common to dynamic options of a specific type.

There are two classes of dynamic options:

- **Interval Set**: Used to represent dynamic option types that provide a set of intervals that will be included in a calculation.
- **Usage Event**: Used to represent dynamic option types that provide a specific period of time during which a dynamic option is applicable.

Interval Set dynamic option types also define key information about the interval data they represent, including:

- **Unit of Measure**: Identifies the quantity that Interval Set dynamic options and Interval Set dynamic option events based on this type are measuring.
- **Interval Size**: Defines the size of the intervals for Interval Set dynamic option events of this type, represented as hours:minutes:seconds (HH:MI:SS).

Configuring Dynamic Option Types

This portal is used to display and maintain a Dynamic Option Type.

Refer to Understanding Dynamic Option Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Dynamic Option Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Dynamic Option Type List**: displays all of the Dynamic Option Types so you can choose the one you want to display in more detail
- **Dynamic Option Type**: shows the specific configuration for the selected Dynamic Option Type

Contact Types

Understanding Contact Types

Contact types define the properties of a class of entities (businesses, persons).

Configuring Contact Types

This portal is used to display and maintain a Contact Type.

Refer to Understanding Contact Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Contact Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Contact Type List**: displays all of the Contact Types so you can choose the one you want to display in more detail
- **Contact Type**: shows the specific configuration for the selected Contact Type

Bill Cycle

Understanding Bill Cycle

Bill cycle identifies window period when a customer is going to be billed and when the bill determinants are going to be calculated.

Configuring Bill Cycle

This portal is used to display and maintain a Bill Cycle.

Refer to Understanding Bill Cycle for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **Bill Cycle**.

The following zones may appear as part of the portal's **Main** tab page:

- **Bill Cycle List**: displays all of the Bill Cycles so you can choose the one you want to display in more detail.
- **Bill Cycle**: shows the specific configuration for the selected Bill Cycle.
- **Bill Cycle List**: lists the bill cycle schedules of the current bill cycle.

Chapter 14

Defining Time of Use Mapping Options

Time of Use Groups

Understanding Time of Use Groups

Time of Use (TOU) Groups are groups of TOUs that are associated to TOU templates that limit the TOUs available for use in a TOU schedule. These TOUs will be available as the default, holiday, and schedule TOUs within the template.

Each TOU in the group is given a priority. The priority is important in a few ways:

- The TOU Overlay 360 Degree zone uses these priorities to decide which TOUs will be rendered as a unique shade and which will fall into a category of "other" when the maximum number of distinct TOUs to graph has been reached.
- These priorities are available for use in customized logic.

Configuring Time of Use Groups

This portal is used to display and maintain a TOU Group.

Refer to Understanding Time of Use Groups for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **TOU Group**.

The following zones may appear as part of the portal's **Main** tab page:

- **TOU Group List:** This zone lists all TOU Group records. Broadcast a record to display the details of the selected record.
- **TOU Group:** This zone provides information about the selected TOU Group.

Time of Use Map Templates

Understanding Time of Use 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 template.

Attributes used to define TOU map templates include the following:

- **TOU Group**: the TOU group used by the map template Default TOU: the default TOU for the 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.
- **Work Calendar**: the work calendar used to identify holidays. On each holiday the Holiday TOU will be used. For additional information see Defining Work Calendar in the Oracle Utilities Application Framework *Administrative User Guide*.
- **Holiday TOU**: the TOU used for holidays (from the TOU Group)
- **Holiday Template**: the TOU map template used for holidays (if applicable)
- **Interval Size**: the size of the intervals for TOU map data created from the map template, represented as hours:minutes:seconds (HH:MI:SS).
- **TOU Schedules**: date ranges (including month, day, and time ranges) and which TOUs

Refer to Understanding Referencing Master Data by Identifiers for information on how admin configuration can reference a TOU map by a TOU template to ease migration of data between environments.

TOU Map Template Interval Size

TOU map templates can also specify an interval size (in seconds-per-interval, or SPI). 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 SPI is divisible by the template's SPI. For example, a 60 minute template can be used to generate TOU data for TOU maps with SPIs of 60 minutes, 15 minutes, 5 minutes, etc. This means separate map templates are not needed for every SPI.

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 in two ways. Both options require that the template references a Work Calendar that identifies each of the holidays throughout the year. In addition they require either:

- A Holiday TOU that will be used for each holiday for the duration of that holiday (e.g. Off Peak)

- A Holiday TOU Map Template that defines the TOUs that should be used for holidays that fall during different seasons within the year (e.g. Off Peak Summer, Off Peak Winter)

Important Time of Use Template System Events

The time of use template supports the following business object algorithm system events:

- **Derive TOU**: receives a date time and determines the TOU code for that date time based on the configuration of the time of use template schedule. See the Derive Time Of Use For Date Time (D2-DERTOU-DT) algorithm type as an example.

Configuring Time of Use Map Templates

This portal is used to display and maintain a TOU Map Templates.

Refer to Understanding Time of Use Map Templates for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **TOU Map Template**.

The following zones may appear as part of the portal's **Main** tab page:

- **TOU Map Template List:** This zone lists all TOU Map Template records. Broadcast a record to display the details of the selected record.
- **TOU Map Template:** This zone provides information about the selected TOU Map Template.

Time of Use Map Types

Understanding Time of Use Map Types

TOU Map Types define important properties of TOU maps of the type, including the interval size and the valid TOU map templates.

Attributes used to define TOU map types include the following:

- **Time Zone**: the time zone to be used when generating the TOU map data. Refer to Multiple Time Zone Support for more information about how time zones impact TOU map data in multiple time zone environments.
- **Interval Size**: the size of the intervals for TOU map data created from maps of this type, represented as hours:minutes:seconds (HH:MI:SS). The interval size cannot be larger than the interval size defined on the Default TOU Map Template or any of the Override TOU Map Templates.
- **Default TOU Map Template**: the default TOU map template used by maps of this type
- **Override TOU Map Templates**: one or more TOU map templates that can be used as an override on TOU maps of this type.

TOU Map Type Interval Size

The SPI of a TOU map must divide evenly into the SPI of any measuring component that uses the map (because the system joins the date/time of the measurement 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 measuring component. However, it is not OK to have a 60 minute TOU map used with a 15 minute measuring component because the join will miss 3 out of 4 measurements.

However, it is important to note that the TOU mapping process is at its most efficient when the measurement data that is being mapped is of the same interval size as the TOU schedule it is being mapped against. When there are differences in the interval size the process must first convert the measurement data into the appropriate interval size for the TOU Map prior to applying the TOU Map.

This means that for each TOU Map Template you should have sufficient TOU Map Types to cover the various interval sizes that will be supported by your measurement data. For example, if you have measuring components with interval sizes of 15 minutes, 30 minutes, and 60 minutes then for each TOU Map Template there should be TOU Map Types with interval sizes of 15 minutes, 30 minutes, and 60 minutes.

Default and Override TOU Map Templates

While most TOU maps will use the TOU map template defined on the TOU map type, TOU maps also support a fallback/override pattern used in other areas of the system.

- A TOU map's TOU map type defines the default (or "fallback") 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 "fallback" template.
- An individual TOU map can have an override template. If the TOU map doesn't have an override template, the fallback template defined on the TOU map type is used to generate the map's TOU data.

Important Time of Use Map System Events

The TOU Map business object that is associated to a given TOU map type supports a special system event that is used in the generation of TOU map data:

- **Create TOU Map Data**: receives a date range and for that date range it will create the appropriate TOU map data for the TOU map. See the Create TOU Map Data (D2-CRETMD-CT) algorithm type as an example.

Configuring Time of Use Map Types

This portal is used to display and maintain a TOU Map Types.

Refer to Understanding Time of Use Map Types for more information.

You can access the portal by selecting **Admin**, then **Usage**, then **TOU Map Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **TOU Map Type List:** This zone lists all TOU Type records. Broadcast a record to display the details of the selected record.
- **TOU Map Type:** This zone provides information about the selected TOU Map Type.

Chapter 15

Defining Communications Options

Device Event Types

Understanding Device Event Types

Device Events

Device event types define properties common to specific types of events.

Device event types represent different types of events that can take place relative to a device. Examples of device events include power outages, power restoration, tampering alerts, and other events.

Device event types can be defined by the following attributes:

- **Standard Event Name:** the "standard" name of the event type. Device vendors may have their own specific names for device events. Only a single active device event type may be mapped to a given standard name at any time.
- **Device Event Category:** a category (defined as an Extendable Lookup) used to group device event types.
- **Reporting Category:** a category used to group device event types for reporting purposes.
- **Activity Type:** the activity type for activities created for device events of this type.

Device Event Mapping

The first step to device event type configuration is defining the list of standard event names that will be processed by the system. This is done by populating the Standard Event Name extendable lookup (D1-StdEventNameLookup). More information about this extendable lookup can be found in About Device Events.

With the fully defined list of Standard Event Names each head-end specific event name (which is also called an external event name) that the system will receive from a given head-end should be mapped to a standard event name. This mapping is configured using a device event mapping extendable lookup. Each head-end system should have its own extendable lookup to define event name mapping in order to prevent possible conflicts between mappings.

Each of these business objects should be defined as a child of the parent business object Device Event Mapping (D1-DeviceEventMappingLookup).

For example, head-end systems A and B might both use the same event name, such as the code "1", but this event might need to be mapped to "outage" for head-end system A but "tamper" for head-end system B.

The device event mapping extendable lookup business object is configurable. Each Oracle Utilities Smart Grid Gateway adapter includes a device event mapping lookup business object for the supported head-end system.

Lastly, each device event type is associated to a standard name. This means that as each device event that is received will go through the following mapping steps:

1. The head end specific event name is mapped to a standard event name using the head end specific device event mapping extendable lookup
2. The standard event name is mapped to a device event type
3. The device event type is used to select the appropriate device event business object

Device Event - Additional Processing

Certain device event business objects can be used to update master data as a result of device events received into the system. For example, when a utility's field worker arms a meter, the resulting device event can trigger an update to the device's arming status in the Oracle Utilities application. This processing is initiated by the Execute - Additional Processing system event

algorithm during the Additional Processing status of the business object's lifecycle.

The following base package business objects are configured to execute additional processing:

If a standard device event requires additional processing, the algorithms that execute the processing should be specified in the Additional Processing Algorithms list on the Standard Event Name extendable lookup for the event. The base package includes the additional processing algorithm Arm Meter (D1-ARM-METER).

Only a subset of base package device event business objects are configured to execute additional processing

Description	Business Object Name
Device Event Communication Response	D1-DeviceEventComResp
Standard Device Event	D1-StandardDeviceEvent

If additional processing is required for any business objects not listed above it can easily be added by configuring the algorithm Executer - Additional Processing System Event (D1-EXTADDPRC) in the Additional Processing status.

Reader Remarks

Reader remark types define properties common to specific types of reader remarks.

Reader remarks are a type of device event used to capture and/or record specific events or circumstances encountered when a meter reader is manually reading scalar meters. Reader remark types represent the different types of remarks that meter readers can record. Examples of reader remark types include evidence of tampering, broken seals, damaged meter, dog on premises, and other notices.

When creating a new device type there are the following options:

Name	Details	Business Object
Reader Remark Type	Provides the configuration for a reader remark type	D1-ReaderRemarkType

Field Activity Remarks

Field activity remark types define properties common to specific types of field activity remarks.

Field activity remarks are a type of device event used to capture and/or record specific events or circumstances encountered when a field worker is performing field work at a service point. Field activity remarks represent can represent situations that are found as well as actions that were taken in the field.

When creating a new device type there are the following options:

Name	Details	Business Object
Device Field Remark Type	Provides the configuration for a field activity remark type	D1-DeviceFieldRemarkType

Configuring Device Event Types

This portal is used to display and maintain a Device Event Type.

Refer to Understanding Device Event Types for more information.

You can access the portal by selecting **Admin**, then **Communication**, then **Device Event Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Device Event Type List:** This zone lists all Device Event Type records. Broadcast a record to display the details of the selected record.
- **Device Event Type:** This zone provides information about the selected Device Event Type.

Activity Types

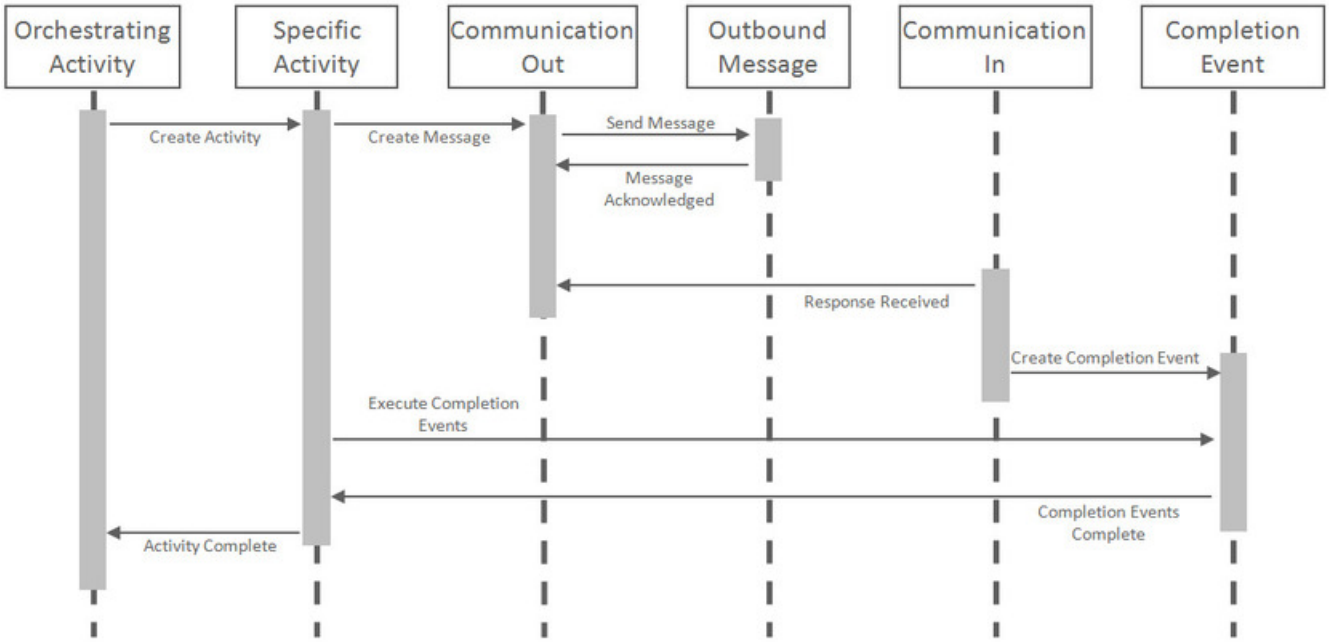
Understanding Activity Types

Activities are records of a communication or event related to a device, measuring component, service point or other entity in the system. Examples of activities include smart meter command request, field activities, meter read downloads (for manually read meters) or the combined event of a "last gasp" and "power up" message sent by devices when they detects they experience an outage.

Each type of activity is assigned to an activity type category, please refer to About Activities for more information about the activity type categories supported by base product.

Activities Orchestrate Communication

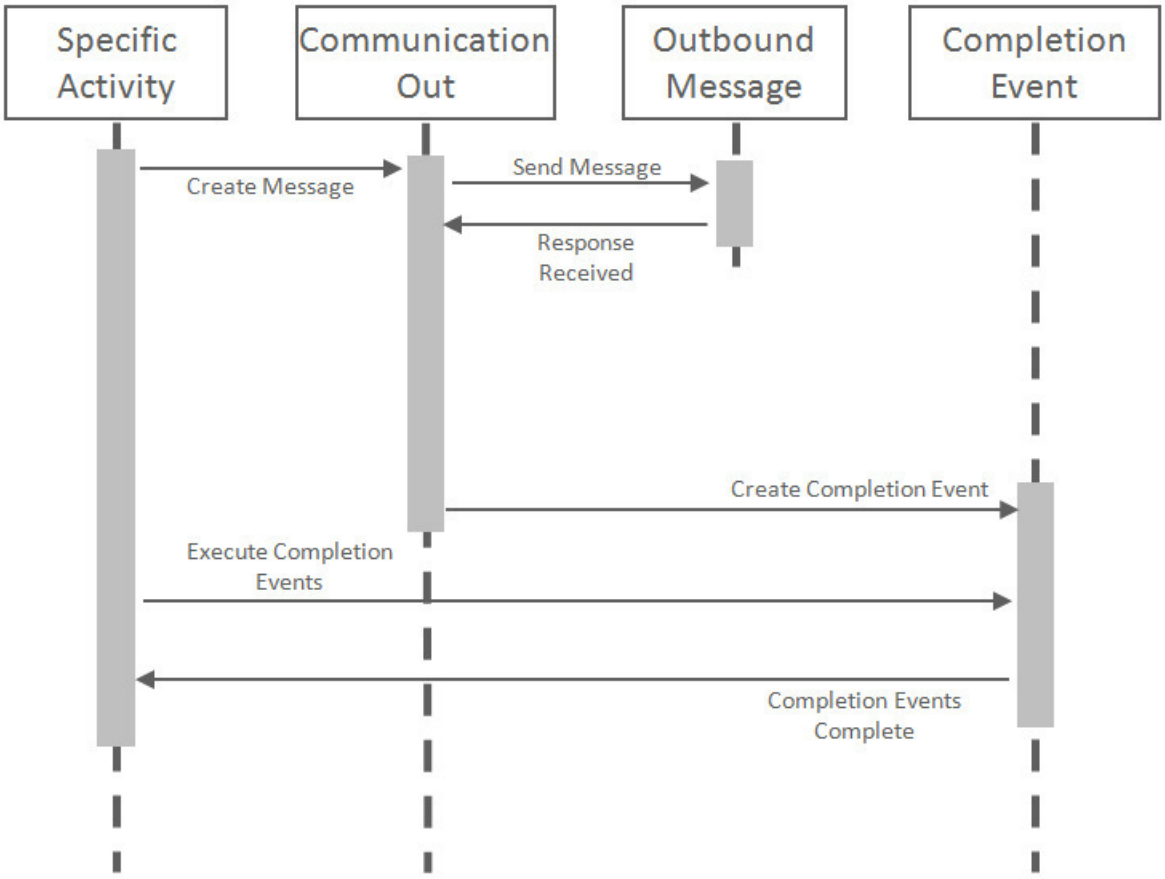
Certain types of activities such as request orchestrators, field activities, and command requests coordinate a large number of child transactions that represent the communication to and from an external application. The below diagram depicts a two way communication with an external system:



Each object in the sequence diagram has a distinct set of duties within the context of the communication:

- Orchestrating Activity: controls the overall intent of the communication. For example it may "Start Service" which would include initiating one-to-many specific activities to install the meter and begin the flow of the metered commodity to the service point.
- Specific Activity: can be initiated from an orchestrating activity or directly. These activities represent a single task to be carried out such as installing a meter or a remote disconnect smart meter command.
- Communication Out: orchestrates the communication to the external application and provides robust handling for any errors that might occur during that communication.
- Outbound Message: represents the message payload sent to the external system and the synchronous response.
- Communication In: orchestrates the handling of an asynchronous or unsolicited response from an external system.
- Completion Event: carries out the results of the communication. For example, in the case of a remote connect it would create the appropriate on off history entry for the device's installation event.

The below diagram involves many of the same objects but instead represents a one-way communication with an external system. All objects maintain the same duties described above with the addition of the communication out handling the synchronous response which contains the result of the message.



Important Activity Type System Events

The activity type supports several business object algorithm system events that relate to calculating the consumption for measuring components of that type:

- **Customer-Device Compatibility**: receives information about the activity being analyzed and returns an indication of whether the customer and device are compatible. This system event is primarily used in service order management. See algorithm type Ensure Device-Usage Calculation Group Compatibility (D2-ENSDVCCMP) as an example.
- **Override Device / Task**: receives information about the activity such as whether the device is installed at a service point, whether the service point is connected, where it is disconnected (if applicable), and the installation event status override. Based on those inputs it should determine the output device configuration type as well as the field activity business object and field task type. This system event is primarily used in service order management. See algorithm type Evaluate Smart Meter Opt-Out for Device Installation (D2-EVSMOPDV) as an example.

Configuring Activity Types

This portal is used to display and maintain a Activity Type.

Refer to Understanding Activity Types for more information.

You can access the portal by selecting **Admin**, then **Communication**, then **Activity Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Activity Type List:** This zone works differently than the typical zone that list types in that it displays both those activity types that have been configured as well as those activity types that have yet to be configured. Broadcast a record to display the details of the selected record.
- **Activity Type:** This zone provides information about the selected Activity Type

Communication Types

Understanding Communication Types

Communication types define properties common to a specific type of communication. This single admin object covers both communication in and communication out types.

Communication types include types of communications between an application and an external system such as a head-end system or the work management system. These are outbound communications such as smart meter command requests or field activities as well as inbound communications such as a message response that indicates the results of a request.

Communication types typically provide configuration around exception handling:

- **To Do Type and Role**: identifies the To Do to be created when an error is encountered
- **Retry Frequency**: defines how often the communication should be retried. As many errors can be due to connectivity the ability to retry provides automatic resolution.
- **Maximum Retries**: provides an upper limit to the number of times a communication can be re-attempted
- Each communication type can have its own unique set of fields, please refer to the embedded help for that communication type for more details on those fields.

Configuring Communication Types

This portal is used to display and maintain a Communication Type.

Refer to Understanding Communication Types for more information.

You can access the portal by selecting **Admin**, then **Communication**, then **Communication Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Communication Type List:** This zone works differently than the typical zone that list types in that it displays both those communication types that have been configured as well as those communication types that have yet to be configured. Broadcast a record to display the details of the selected record.
- **Communication Type:** This zone provides information about the selected Communication Type

Service Task Types

Understanding Service Task Types

Service tasks types define properties common to specific types of service tasks.

Service task types represent types of tasks that can be performed by users of other Oracle Utilities applications, such Oracle Utilities Customer Self Service or Oracle Utilities Network Management System. Examples of service tasks include self service meter reads, in which users enter their own meter reads via the Customer Self Service application, and service issue monitor types used when determining if service investigation is needed for a service point.

When creating a new record, the following options are available:

Name	Details	Business Object
Service Issue Monitor Type	The service issue monitor type defines the parameters for when a service investigative order should be created. It can also optionally define the completion parameters for that indicate the investigative order was successful in its purpose.	D1-ServiceIssueMonitorType
Self-Service Rate Compare Scenario Request Type	A simple description of a rate compare scenario request type.	D2-RateCompareScenarioRqstType
Self-Service Meter Read Task Type	Provides the default settings for meter reads created through self service.	D2-SSMeterReadTaskType

Configuring Service Task Types

This portal is used to display and maintain a Service Task Type.

Refer to Understanding Service Task Types for more information.

You can access the portal by selecting **Admin**, then **Customer**, then **Service Task Type**

The following zones may appear as part of the portal's **Main** tab page:

- **Service Task Type List:** This zone lists all Service Task Type records. Broadcast a record to display the details of the selected record.
- **Service Task Type:** This zone provides information about the selected Service Task Type.

Chapter 16

Smart Grid Gateway Adapters

Smart Grid Gateway Adapters and On-Premises Implementations

NOTE: Unless otherwise noted, the contents of this section apply to on-premises implementations of Oracle Utilities Smart Grid Gateway only, and do NOT apply to Oracle Utilities cloud services. See Smart Grid Gateway Adapters and Oracle Utilities Cloud Services for more information about implementing Smart Grid Gateway Adapters with Oracle Utilities cloud services.

Oracle Utilities Smart Grid Gateway (SGG) Adapters support communication with various third-party head-end systems. SGG uses Oracle Service Bus (OSB) and Oracle Business Process Execution Language (BPEL) to facilitate communication with the head-end systems. Most if not all of the functionality shown below is available in each SGG Adapter.

Measurement Data and Device Event Loading: Data parsing and transformation via Oracle Service Bus from head-end system format into the Oracle Utilities Service and Measurement Data Foundation unified format for measurement data and device events.

Measurement Data and Device Event Processing: Configurable mapping for head-end system status codes and device event names to Oracle Utilities Service and Measurement Data Foundation standard values.

Smart Meter Command Processing: Sending/receiving messages to/from third-party applications to initiate smart meter commands from SGG. Most head-end systems support the following types of commands and communications:

- **Device Status Check:** Business objects and BPEL processes to support issuing device status check commands.
- **Meter Commissioning**: Business objects and BPEL processes to support issuing meter commissioning commands (including registration and installation commands).
- **Meter Decommissioning**: Business objects and BPEL processes to support issuing meter decommissioning commands.
- **Meter Retirement:** Business objects and BPEL processes to support issuing meter retire (deregistration) commands.
- **On-Demand Read**: Business objects and BPEL processes to support issuing on-demand read commands.
- **Remote Connect**: Business objects and BPEL processes to support issuing remote connect commands.

- **Remote Disconnect**: Business objects and BPEL processes to support issuing remote disconnect commands.

This information in this guide describes how to configure, extend, and test the SGG Adapter for individual head-end systems. The following head-end systems each have a specific SGG Adapter that can be configured: Itron OpenWay, Landis +Gyr, Networked Energy Services, MV90 for Itron, Sensus RNI, and Silver Spring Networks. Customers also can create their own customized adapter for using Smart Grid Gateway with a specific head-end system. See Creating a Custom Adapter for Smart Grid Gateway for more information.

Configuring Smart Grid Gateway Adapters

This section describes how to configure Oracle Utilities Smart Grid Gateway Adapters for each supported head-end system. Use the cross-references below to go to the adapter of interest.

Itron OpenWay

Landis+Gyr

Networked Energy Services

MV90 for Itron

Sensus RNI

Silver Spring Networks

NOTE: This section applies to on-premises implementations of Oracle Utilities Smart Grid Gateway only. This section does NOT apply to Oracle Utilities cloud services. See Smart Grid Gateway Adapters and Oracle Utilities Cloud Services for more information about implementing Smart Grid Gateway Adapters with Oracle Utilities cloud services.

Itron OpenWay

The Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay supports communication with the Itron OpenWay application, including measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, status check, and on-demand read commands. The following table describes the attributes of the adapter:

Attribute	Details
Currently Supported Versions	7.0 Itron OpenWay Operations Center SR 4.5
Protocol	Proprietary
Market(s)	Worldwide
Architecture	RF Mesh

Itron OpenWay Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, and Oracle Utilities Application Framework (OUAF) objects that are supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data and Device Event Loading

The initial measurement data load and device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway (SGG) service format. Payloads contain measurements and meter events in some head-end specific format. OSB then places each service call into a Java Message Service (JMS) queue within the

Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel. A service then creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format.

Measurement and device event data must be requested from the Itron OpenWay head-end system via Scheduled Read commands. See Scheduled Read Commands for more information.

Scheduled Read Commands

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Itron OpenWay.

Measurement and device event data must be requested from the Itron OpenWay head-end system via Scheduled Read commands. Schedule read commands are used to request meter reads and/or device events for a group of devices for a specified date and time. Scheduled read commands can be created such that recurring requests based on the same command parameters are sent to the head-end system at regular intervals. For example, a command could be configured to request daily (24 hour) scalar meter readings once per day for a specified group of devices. Another command could be configured to request weekly (168 hours) device events once per week for a group of devices.

Scheduled read commands should be created for all devices from which meter readings and/or device events are to be uploaded via Oracle Utilities Smart Grid Gateway. Separate scheduled read commands can be created for specific groups and for specific types of data. For example, for a given group of scalar devices, you might create separate commands to request scalar usage data and to request device events from the same group (both of which can be based on different recurrence patterns). The specifics concerning request specifics and recurrence patterns should be based on the requirements of the implementation.

Attributes used to define a schedule read command include the following:

- **Status:** the status of the activity
- **Schedule Read Type:** the type of scheduled read
- **Schedule Information:** information about the schedule for the read, including:
  - **One Time Read:** indicates if the request is a one-time request. One-time requests have a defined date/time range during which request and recurrences can take place.
  - **One Time Start Date Time:** specifies the start time for one-time requests (applicable only if **One Time Read** is set to “Yes”)
  - **One Time End Date Time:** specifies the end time for one-time requests (applicable only if **One Time Read** is set to “Yes”)
  - **First Daily Measurement Time**: Indicates starting time for measurements for each day. This field, along with the value from **IMD Length** , will be used to determine when interrogation requests should be made. For example, if the **First Daily Measurement Time** is set to 08:00AM and **IMD Length** is set to 12 hours (12:00:00) then there would be two requests made per day:
    - 8:00AM - 8:00PM
    - 8:00PM - 8:00AM
  - **Interrogation Buffer**: This parameter impacts when interrogation requests are sent. It is entered in the format of hours:minutes:seconds. For example, if set to 1 hour (01:00:00) for our above example, the first request of the day for 8:00AM - 8:00PM would be sent to Itron OpenWay at 7:00PM. The buffer time will decrease if the activity is monitored after 7:00PM, so if the monitor process is executed at 7:30PM the buffer time would only be 30 minutes and if the monitor process were not executed until after 8:00PM there would be no buffer time at all.
  - **IMD Length**: Defines the number of hours, minutes and seconds of data that should be retrieved for each request. It is entered in the format of hours:minutes:seconds. This number must be divisible into 24 since it will determine how many requests will be made per day (i.e. if set to 8 hours (08:00:00) there will be 3 requests per day).

- **Disable Extended IMD Length Recovery**: By default the schedule read activity will attempt to "catch-up" when interrogation requests have been missed by requesting the entire period from the last interrogation request made. Setting this parameter to "Yes" will turn off this functionality and interrogation requests will only be made in for the exact amount of time defined in the **IMD Length**
- **Range of Recurrence:** indicates how to define request recurrences. Valid values include “Maximum Recurrence”, “No End Date”, and “Recurrence End Date”
  - **Maximum Recurrence**: The activity will make a set number of requests, once the set number has been reached the activity will expire. When selected, **Maximum Recurrences** must also be provided.
  - **No End Date**: The activity will never expire and will continue to make requests until it is manually transitioned to inactive.
  - **Recurrence End Date**: The activity will continue to make requests until a specified date. When selected, **Recurrence End Date Time** must also be provided.

    NOTE: When “No End Data” is selected, the recurrences of the request continue until a user manually changes the status of the command.

- **Maximum Recurrence:** defines the maximum number of recurrences before the command’s status is changed to “Inactive”.
- **Recurrence End Date Time:** The date after which no more recurrences are attempted. After this date and time, the command’s status is changed to “Inactive”.

- **Group:** the group of devices used by requests for the command
  - **Group Type:** indicates the type of device group to use for the command. Valid options include Application Group or Configuration Group.
  - **Application Group/Configuration Group Name:** defines the specific group (based on the group type).

    NOTE: Groups referenced by Scheduled Read commands and the devices that belong to each are defined in Itron OpenWay head-end system. Each group to be used with these commands must also be defined in either the “Itron — Application Group Lookup” or “Asset Specification” extendable lookup. See the Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay Configuration Guide for more information about these extendable lookups.

- **Default Read Parameters:** specifies details concerning the data to be requested. The read parameters define the exact data that will be requested from the meters. Please consult the Itron OpenWay documentation for the specifics of each parameter.
  - **Interrogation Window Hours:** the number of hours used to calculate when to submit requests. For example, to submit requests once per day, this should be set to 24. To submit requests once per week (7 days), this should be set to 168. The current date time (or the Latest Request End Date Time) plus this equals interrogation window end date time.
- **Override Read Parameters**: an alternative set of parameters that can be defined. To leverage this group each parameter must be provided, no parameter is allowed to be left blank. This set of parameters will be used when the request end time is equal to the **First Daily Measurement Time**. For example, a schedule read with a **First Daily Measurement Time** of 12:00AM and an **IMD Length** 12 hours (12:00:00) would make the following requests each day:
  - 12:00AM - 12:00PM
  - 12:00PM - 12:00AM

  The **Override Read Parameters** would be used for the 2nd request where the request end time of 12:00AM matches the **First Daily Measurement Time** of 12:00AM. If an extended request is made to "catch up" (i.e. it makes a request that is larger in duration than the configured **IMD Length**), and the request start and request end date time includes the **First Daily Measurement Time**then the **Override Read Parameters** will be used.

- **Outbound Communication Exception Handling Overrides**: parameters used to control retry attempts and expiration for the outbound communication. These parameters take precedence over their outbound communication counterparts
- **Last Read Details:** details concerning the most recent read request sent for the command. This information is used to calculate the time for the next request (based on the **Interrogation Window Hours** parameter).
  - **Latest Request Start Date Time:** the start date and time of the last request
  - **Latest Request End Date Time:** the end date and time of the last request
  - **Recurrence Count:** the number of recurrences of the last request

About Ad Hoc Requests

While scheduled read commands are typically configured to create recurring requests for meter reads and device events, users can also submit ad hoc requests for a specific date and time range based on the parameters of a currently active scheduled read command. This is useful if meter read data or device events are needed before the next request for a command would be submitted based on the command's recurrence settings.

Ad hoc requests are created as separate one-time request commands, with a start and end time based on parameters supplied by the user when initiating the ad hoc request. The group and read parameters for ad hoc requests are the same as those for the command used to send the ad hoc request.

Error Handling

If errors occur when processing scheduled read commands, errors are logged as follows:

- If there is a single error it will create a single log entry
- If there is more than one error, the log entry will perform a count of devices and create a single error message stating the number of devices that were in error.

Initial Measurements

The usage data received from the AMI head-end system as a file in Itron OpenWay XML format is loaded into Oracle Utilities as initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. **SGG-D8-ITRONXML-BASE** contains components responsible for "actual" processing of incoming data. It should not be modified during configuration. This project can be upgraded without affecting the customization and environment settings added to SGG-D8-ITRONXML-CM.
2. **SGG-D8-ITRONXML-CM** allows for customization and simplifies future upgrades.

The runtime configuration settings for the SGG-D8-ITRONXML-CM project are stored in the xquery file EnvironmentSettings.xq. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify "true" for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	

Element	Description	Valid Values
destinationRootElementScalar	Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval.	
destinationRootElementEvent	Holds the name of inbound web service for the device event seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false
filterUsage	Determines if usage should be filtered.	true false
filterRegisterSource	Determines if source register data should be filtered.	true false

Publishing Initial Measurement Data

The SGG adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Initial Measurement Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D8-ITRONXML-CM OSB project are used in publishing measurement data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Initial Measurement Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Initial measurement data is published in the “native” initial measurement data format (the format of the initial measurement seeder business object). This format includes normalized unit of measure and condition codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published through this feature is published prior to validation, estimation, and editing (VEE) processing. In addition, filtering *cannot* be applied to data published through this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads.	DataRakerQueue

Parameter	Description	Default Value
	This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Initial Measurement Data

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **Itron - UOM Code to Standard UOM Mapping** extendable lookup (D8-HeadendUOMLookup) are passed into the system for processing.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service for the project. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Prioritized Initial Measurement Processing

The SGG adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to “Real Time” (D1RT). In addition, initial measurements received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Device Events

Itron OpenWay can send real-time events such as meter outage alarms to Oracle Utilities Smart Grid Gateway. These events are referred to as exceptions in the Itron architecture, and are sent to the BPEL ExceptionSubscriberService service by Itron

OpenWay without requiring an explicit request from Oracle Utilities Smart Grid Gateway. Exceptions are loaded as device events data and provided to the edge applications.

The required functionality is delivered in the base product as two OSB projects:

1. **SGG-D8-EXCEPTION-BASE** contains components responsible for "actual" processing of incoming data. It should not be modified during configuration. This project can be upgraded without affecting the customization and environment settings added to the SGG-D8-EXCEPTION-CM project.
2. **SGG-D8-EXCEPTION-CM** allows the customization and simplifies the future upgrades.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the device event data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElement	Holds the name of inbound web service for the device event seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false

Publishing Events

The Itron OpenWay adapter can be configured to publish device events for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Device Event Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D8-EXCEPTION-CM OSB project are used in publishing device events data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Device Event Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Device event data is published in the "native" device event data format (the format of the device event seeder business object). This format includes normalized device event codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: In addition, filtering can NOT be applied to device events published via this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Events

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **Itron - Device Event Mapping** extendable lookup (D8-DeviceEventMappingLookup) are passed into the system for processing.

Prioritized Device Event Processing

The SGG adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Configuring Payload Processing Notifications

By default, Payload Statistics, Payload Error Notification, and Payload Summary notification activities are sent to the Notification Queue within Oracle Service Bus. Smart Grid Gateway adapters can be configured to restrict sending these activities by defining optional JCA transport dynamic properties in the appropriate OSB project. The following optional JCA transport dynamic properties can used to control if these notification activities are sent:

- **SGG_SEND_STATISTICS**: Controls sending Payload Statistic activities (true: send, false: do not send)
- **SGG_SEND_ERROR**: Controls sending Payload Error Notification activities (true: send, false: do not send)
- **SGG_SEND_SUMMARY**: Controls sending Payload Summary activities (true: send, false: do not send)

When these properties are set to false, the corresponding payload notification will not be sent to the JMS Queue. Note these properties are not populated by default, and default to true if omitted.

To create these properties:

1. Navigate to the SGG-D8-EXCEPTION-CM project in the Oracle Service Bus Console.
2. Expand the **Proxy Services** folder.
3. Select the **InboundProxyService** proxy service (not the Pipeline or WSDL).
4. Click **Create** to create a new session.
5. Select **Transport Details**.
6. Click the plus sign in the **Dyanmic EndPoint Properties** section. An empty row will appear.
7. Enter the property you wish to define in the **Property** column.
8. Enter "false" in the **value** column.
9. Repeat steps 6-8 for each property you wish to define.
10. Click **Activate** to save and activate your changes.

NOTE: These properties are only used if the SGG_EXT_ID_PATTERN is not present or the regular expression captures nothing. The SGG_EXT_ID_PATTERN will override these properties when the regular expression captures group data.

Base Package Business Objects

The Itron OpenWay adapter base package includes the following initial measurement and device event business objects:

Business Object Name	Description
D8-InterrogateByGroup	Itron - Interrogate By Group Performs a Schedule Read for a group of devices (either Application or Configuration) in the Itron OpenWay API.
D8-InterrogateByGroupResult	Itron - Interrogate by Group Result This BO is the inbound asynchronous response to an Interrogate By Group (Schedule Read) outbound communication.
D8-InitialLoadIMDInterval	Itron - Initial Load IMD - Interval Used when loading Itron OpenWay interval measurements into the system for the first time.
D8-InitialLoadIMDScalar	Itron - Initial Load IMD - Scalar Used when loading Itron OpenWay scalar measurements into the system for the first time.
D8-ScheduleRead	Itron - Schedule Read

Business Object Name	Description
	Requests IMD and Event data from the Itron OpenWay Network. This data will later be returned and placed in files to be uploaded.

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/ decommission, measurement data, an on-demand read, or another type of request that the head-end system supports. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related web service from the head-end system. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

Communication Flows

The table below lists the communications created for each Itron OpenWay command:

Command	Outbound Communication	Inbound Communication	Completion Event
Remote Connect	Itron - Reconnect Meter	Itron - Reconnect Meter Result	Connect Device Completion Event
Remote Disconnect	Itron - Disconnect Meter	Itron - Disconnect Meter Result	Disconnect Device Completion Event
Device Commissioning	Itron - Add Meter Definition		Device Commissioning Completion Event
Device Decommissioning	Itron - Deregister Meter		Device Decommissioning Completion Event
On-Demand Read (Scalar)	Itron - Contingency Read (Scalar)	Itron - Contingency Read Result (Scalar)	Create IMD Completion Event
On-Demand Read (Interval)	Itron - Contingency Read (Interval)	Itron - Contingency Read Result (Interval)	Create IMD Completion Event
Scheduled Read (Scalar)	Itron - Interrogate by Group (Scalar)	Itron - Interrogate by Group Result (Scalar)	Create IMD Completion Event
Scheduled Read (Interval)	Itron - Interrogate by Group (Interval)	Itron - Interrogate by Group Result (Interval)	Create IMD Completion Event
Device Status Check	Itron - Ping by Endpoints (Status Check)	Itron - Get Ping by Endpoints Result	

Device Communication Base Package Business Objects

The Itron OpenWay Adapter base package includes the following communication business objects:

Business Object Name	Description
D8-AddMeterDefinitions	Itron - Add Meter Definition (Commission)
D8-DeregisterMeter	Itron - Deregister Meter (Decommission)
D8-DetectLoadSideVoltageByMtr	Itron - Detect Load Side Voltage
D8-DetLoadSideVoltageMtrRslt	Itron - Detect Load Side Voltage Result
D8-DisconnectMeter	Itron - Disconnect Meter (Remote Disconnect)
D8-DisconnectMeterResult	Itron - Disconnect Meter Result
D8-PingByEndpoints	Itron - Ping By Endpoints (Device Status Check)
D8-PingByEndpointsMDResponse	Itron - Multi-Device Ping Response
D8-PingByEndpointsMultiDevice	Itron - Ping By Endpoints Multi-Device

Business Object Name	Description
D8-PingByEndpointsResponse	Itron - Ping By Endpoints Response
D8-ReadDisconStateByMtr	Itron - Read Disconnect State
D8-ReadDisconStateMtrRslt	Itron - Read Disconnect State Result
D8-ReadInterval	Itron - Contingency Read (Interval)
D8-ReadResult	Itron - Contingency Read Result
D8-ReadScalar	Itron - Contingency Read (Scalar)
D8-ReconnectMeter	Itron - Reconnect Meter (Remote Connect)
D8-ReconnectMeterResult	Itron - Reconnect Meter Result

Itron OpenWay Event Data Mapping

The Itron OpenWay event file format maps as follows into the business object, D1-DeviceEventMappingLookup:

Itron OpenWay Flat File Field	Device Event Seeder BO Element	Comments
Transaction ID (from Header record)	External Source Identifier	This is the file name.
Device Identifier	External Device Identifier	
Event Name	External Event Name	
Event Creation Date/Time	Event Date/Time	
Device Type	External Device Type	This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form): Meter Collector Router
Service Location ID	External Service Location ID	
Communication Module Serial Number	External Communication Module Identifier	
Event Category ID	External Event Category	
Event Severity	External Event Severity	Valid values include (although the element itself is free-form): Alert Information
Status Value	External Status Value	This represents additional information that relates to the event itself.
Status Date/Time	External Status Date/Time	The date & time at which the additional information referenced above had occurred.

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch or Real-time)

- The corresponding message senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgment and response messages are sent and received validating that commands have been transmitted.

Outbound Message Type	Description
D8ITRECO	D8 Itron Remote Connect

Inbound / Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The Oracle Utilities Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway adapter for Itron OpenWay includes the following inbound web services:

Inbound Web Service	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Used for upload of device events. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-DeviceStatusCheck	Device Status Check This service is invoked by the integration layer to instantiate a Device Status Check command.
D1-InitialLoadIMD	Used by OSB to instantiate an IMD The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.
D1-RemoteConnect	Remote Connect This service is invoked by the integration layer to instantiate a Remote Connect command.

Inbound Web Service	Description
D1-RemoteDisconnect	Remote Disconnect This service is invoked by the integration layer to instantiate a Remote Disconnect command.
D8-DetLoadSideVoltageMtrRslt	Itron - Detect Load Side Voltage by Meter Result
D8-DisconnectMeterResult	Itron - Disconnect Meter Result Service
D8-InterrogateByGroupResult	Itron - Interrogate By Group Result XIA
D8-PingByEndpointsMDResponse	Itron - Ping By Endpoints Multi-Device Response XAI Inbound
D8-PingByEndpointsResponse	Itron - Ping By End Response inbound web service
D8-ReadDisconStateMtrRslt	Itron - Read Disconnect State by Meter Result Service
D8-ReadResult	Itron - Contingency Read Result
D8-ReconnectMeterResult	Itron - Reconnect Meter Result Service

Message Senders

Message senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of message senders you need to create is based on the types of messages the system is designed to accept.

The Oracle Utilities Smart Grid Gateway adapter for Itron OpenWay includes the following message senders:

Message Sender	Description
D8-RemoteCon	D8 Remote Connect

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to Itron OpenWay format, invoking process callouts and invoking the remote endpoint to trigger the device events. BPEL modules are divided into “Composites,” or collections of business logic.

OnDemandRead Composite Process — The OnDemandRead composite invokes a sequence of web methods that call the head-end system to retrieve meter reading data and send it back to the OUAF layer. In the case of Itron OpenWay, this is accomplished using the ContingencyReadByEndpoints/GetContingencyReadByEndpointsResult services defined in the Data service WSDL.

ConnectDisconnect Composite Process — This composite is responsible for triggering the Connect and Disconnect events on the head-end system. A second, asynchronous reply sends the results back into the OUAF layer when the head-end system signals a change in the status of the device. The Itron OpenWay Adapter uses the ReconnectMeter/ GetReconnectMeterResult web service pair to connect, and the DisconnectMeter/GetDisconnectMeterResult web service pair to disconnect. Each of these services is defined in the Control service WSDL.

CommissionDecommission Composite Process — Commissioning and Decommissioning of devices, sometimes referred to as “Provisioning,” is handled by the CommissionDecommission composite. Commissioning invokes the AddMeterDefinitions web service and Decommissioning uses the DeregisterMeters web service. Both are defined in the Provisioning service.

DeviceStatusCheck Composite — This composite uses the PingByEndpoints and GetPingByEndpointsResult web services to check the health of a device. These Itron OpenWay services are defined in the Control.Diagnostic service.

Common Composite — The Common composite contains three main classes of operations: Proxies, ProcessCallouts, and utility functions. Proxies are usually simple mediators that forward a web service call to a preset endpoint. In this Adapter

they have two additional roles. First, they interrogate a composite property and determine whether the supported version of head-end software is 3.70 or 3.90. Some of the proxies will also append the callback URL for the StatusChanged service. Proxies are convenient because they allow head-end URLs and security to be set in a single composite. In this case, they also offer a common location to make these checks. ProcessCallouts are points of customization which allow users to modify data and/or initiate some external business process. Utility functions serve as a central location for business logic needed by multiple composites.

Web Services

The following web services are all defined in the Itron OpenWay head-end system:

- **CommissionDecommissionService**
  - **BPEL Process:** CommissionDecommission
  - **Operation:** AddMeterDefinitions
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/<EM_PARTITION>/CommissionDecommission/CommissionDecommissionService
- **CommissionDecommissionService**
  - **BPEL Process:** CommissionDecommission
  - **Operation:** DeregisterMeters
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/<EM_PARTITION>/CommissionDecommission/CommissionDecommissionService
- **ConnectDisconnectService**
  - **BPEL Process:** ConnectDisconnect
  - **Operation:** ReconnectMeter
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/<EM_PARTITION>/ConnectDisconnect/ConnectDisconnectService
- **ConnectDisconnectService**
  - **BPEL Process:** ConnectDisconnect
  - **Operation:** DisconnectMeter
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/<EM_PARTITION>/ConnectDisconnect/ConnectDisconnectService
- **DeviceStatusCheckService**
  - **BPEL Process:** DeviceStatusCheck
  - **Operation:** DeviceStatusCheck
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/<EM_PARTITION>/DeviceStatusCheck/DeviceStatusCheckService
- **OnDemandReadService**
  - **BPEL Process:** OnDemandRead
  - **Operation:** ContingencyReadByEndpoints
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/<EM_PARTITION>/OnDemandRead/OnDemandReadService

Itron OpenWay Web Services

The following table describes Itron OpenWay web services and operations used for the Oracle Utilities Smart Grid Gateway command messaging:

Smart Grid Gateway Command	AMI Adapter Business Objects	Itron OpenWay Web Services	Itron OpenWay Operations
Device Commissioning	D8-AddMeterDefinitions	ProvisioningService	AddMeterDefinitions
Device Decommissioning	D8-DeregisterMeter	ProvisioningService	DeregisterMeters
Remote Connect	D8-ReconnectMeter	ControlService	ReconnectMeter
	D8-DetectLoadSideVoltageByMtr	DiagnosticService	DetectLoadSideVoltageByMeter
	D8-ReadDisconStateByMtr		ReadDisconnectStateByMeters
Remote Disconnect	D8-DisconnectMeter	ControlService	DisconnectMeter
Device Status Check	D8-PingByEndpoints	ControlService	PingByEndpoints
			GetPingByEndpointsResult
On-Demand Read	D8-ReadInterval	DataService	ContingencyReadByEndpoints
	D8-ReadScalar		GetContingencyReadByEndpointsResult
	D1-InitialLoadIMD		InterrogateByGroup
	D1-DeviceEventSeeder		GetInterrogateByGroupResult
		DataSubscriberService (Async Response)	DataArrived
Event Loading	D1-DeviceEventSeeder	ExceptionSubscriberService	ExceptionsArrived

Configuring an Itron OpenWay Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay to communicate with the Itron OpenWay application.

Master Configurations

Master Configurations are sources of global parameter records used by a system implementation. This section describes the master configuration that is specific to Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay. Refer to the Oracle Utilities Meter Data Management / Smart Grid Gateway documentation for more information about other master configurations used by Oracle Utilities Smart Grid Gateway.

Itron Version Master Configuration

This master configuration specifies the version of the Itron head-end system with which the system is communicating. The configuration options are Itron OpenWay 3.70, Itron OpenWay 3.90, and Itron OpenWay 6.10. Only one version can be specified at a time.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the Itron OpenWay application in response to a command request.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Device Event Seeder
D1-DeviceStatusCheck	Device Status Check
D1-InitialLoadIMD	Used by OSB to instantiate an IMD
D1-PayloadErrorNotif	Payload Error Notification
D1-PayloadStatistics	Payload Statistics
D1-PayloadSummary	Payload Summary
D8-DetLoadSideVoltageMtrRslt	Itron - Detect Load Side Voltage by Meter Result
D8-DisconnectMeterResult	Disconnect Meter Result Service
D8-InterrogateByGroupResult	Itron - Interrogate By Group Result XIA
D8-PingByEndpointsMDResponse	Itron - Ping By Endpoints Multi-Device Response XAI Inbound
D8-PingByEndpointsResponse	Itron - Ping By Endpoints Response inbound web service
D8-ReadDisconStateMtrRslt	Itron - Read Disconnect State by Meter Result Service
D8-ReadResult	Itron - Contingency Read Result
D8-ReconnectMeterResult	Itron - Reconnect Meter Result Service

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object
- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

Message Senders

Message senders define the details of how messages are sent to an external system, such as messages containing device command requests. An message sender should be configured for each command.

The following message senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders.

Message Sender	Description
D8-ITRON_IBG	Itron Interrogate By Group
D8-COMM	Itron Commission
D8-DCOMM	Itron Decommission
D8-DLSV	Itron Detect Load Side Voltage
D8-IPBE	Itron Ping By Endpoints

Message Sender	Description
D8-RCONN	Itron Remote Connect
D8-RDCONN	Itron Remote Disconnect
D8-RDSS	Itron Read Disconnect State

Note: The following apply to all of the above message senders:

Main Tab:

- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction:http://xmlns.oracle.com/ouaf/Itron/**<OPERATION>**
- **HTTP Login User**: **<USER_ID>**
- **HTTP Login Password**: **<PASSWORD>**
- **HTTP Method**: POST
- **HTTP URL 1**: http://**<EM_SERVER>**:**<EM_SERVER_PORT>**/soa-infra/services/Itron/**<SERVICE>**
- where:
  - **<OPERATION>**: the operation performed by the message sender (see Operation column in the table above)
  - **<USER_ID>**: the user ID used to log into WebLogic Enterprise Manager
  - **<PASSWORD>**: the password used to log into WebLogic Enterprise Manager
  - **<EM_SERVER>**: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
  - **<EM_SERVER_PORT>**: the port where the WebLogic Enterprise Manager is installed
  - **<SERVICE>**: the service invoked by the message sender (see Service column in the table above)

How to Use Enterprise Manager to Locate the URL for the Message Sender

Follow this procedure to find the correct URL for the command associated with an message sender:

1. Open Enterprise Manager and use the navigation pane to open the dashboard of the service used by the message sender:
2. The top bar of the dashboard contains several buttons and icons. One of these is a “world” icon with a puzzle piece over it. Click this icon to display a list of the WSDLs and endpoint URIs for the service:
3. Click the service’s WSDL URL link to see the WSDL in the browser, or right click and save it to your machine.

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
D8-CONREAD	Itron Contingency Read By Endpoints
D8-INTRGRP	Itron Interrogate By Group
D8-COMM	Itron Commission

Outbound Message Type	Description
D8-DCOMM	Itron Decommission
D8-DSC	Itron Device Status Check
D8-DSCMD	Itron Device Status Check Multi Device
D8-DLSV	Itron Detect Load Side Voltage
D8-RCONN	Itron Remote Connect
D8-RDCONN	Itron Remote Disconnect
D8-RDSS	Itron Read Disconnect State

Note: The following apply to all of the above outbound message types:

- **Business Object**: D1-OutboundMessage (Outbound Message)
- **Priority**: Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the Itron OpenWay head end must be present in your system. If this is not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System - Itron OpenWay:

- **External System**: Itron OpenWay
- **Description**: Itron OpenWay
- **Outbound Message Types**:

Outbound Message Type	Message Sender
D8-INTRGRP	D8-ITRON_IBG
D8-COMM	D8-COMM
D8-DCOMM	D8-DCOMM
D8-DSC	Message sender associated with the Device Status Check Outbound Message Type
D8-DSCMD	Message sender associated with the Device Status Check Multi Device Outbound Message Type
D8-DLSV	D8-DLSV
D8-RCONN	D8-RCONN
D8-RDCONN	D8-RDCONN
D8-RDSS	D8-RDSS

Note: The following apply to all of the above outbound message types:

- **Processing Method**: Real-time
- **Message XSL**: D8-Request xsl
- **Response XSL**: D8-Response.xsl

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment,

outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the Itron OpenWay head end must be present in your system. If this are not present in your configuration, add it. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about creating service providers.

Service Provider - Itron OpenWay:

- **Service Provider**: Itron OpenWay
- **Description**: Itron OpenWay
- **External Reference ID**: Itron OpenWay
- **External System**: Itron OpenWay
- **Our Name/ID in Their System**:
- **AMI Device ID Type**: Internal Meter Number
- **Pre-Commissioning Device ID Type**:
- **AMI Measuring Component ID Type**: Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to a given head-end system.

The following types of processing methods must be configured for the Itron OpenWay service provider. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head-end system.

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.

UOM Translation

UOM mapping processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the Itron OpenWay service provider, based on the requirements of each implementation using the “How to Create OB COMM/Send OB Message” processing method business object (D1–HowToCreateActivityOBComm).

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Registration	D8-AddMeterDefinitions	Commission
Device Decommission	Device Removal	D8-DeregisterMeter	Decommission
Device Status Check	Device Status Check	D8-PingByEndpoints	Device Status Check
Load Check*	Load Check	D8-DetectLoadSideVoltageByMtr	Detect Load Side Voltage by Meter
Multi-Devie Status Check	Multi-Devie Status Check	D8-PingByEndpointsMultiDevice	Read Disconnect State By Meters
On-Demand Read (Scalar)	On-Demand Read (Scalar)	D8-ReadScalar	On Demand Read — Scalar
Remote Connect	Remote Connect	D8-ReconnectMeter	Connect
Remote Disconnect	Remote Disconnect	D8-DisconnectMeter	Disconnect

* The Load Check processing method can be used to ensure that it is safe to connect a meter. The load check processing method is executed during the Connection Ready state of the Remote Connect activity and can be configured to detect the possibility of a load side voltage at the meter, or to check the disconnect meter switch on the meter.

If the Read Disconnect State business object is used as the processing method, you should configure the disconnect switch state values that indicate the state of the switch. These values are defined on the D8-DisconnectSwitchStateLookup extendable lookup, and are then added in the Disconnect Switch State Error Values section of the Itron - Read Disconnect State Result Inbound communication type. See Itron OpenWay Disconnect Switch State Lookup for more information.

Configuring Endpoint URIs

Part of the configuration process is configuring your BPEL composites to work with your head-end system by defining the appropriate Endpoint URIs for each of the commands. The default approach to defining Endpoints URIs is redeployment or reinstallation or the BPEL composites. For example, by default, changing an adapter from using the test harness to a production environment using the actual head-end system requires editing the appropriate installation menu options and redeploying the BPEL composites. The Endpoints URIs defined during installation and deployment for each adapter are listed in the **Smart Grid Gateway Installation and Configuration Worksheets** section of the *Oracle Utilities Smart Grid Gateway Installation Guide*.

You can also use an “Endpoint Override” Domain Value Map (DVM) to override Endpoint URIs defined during deployment and installation. This DVM allows defining specific keys that provide an alternate URL that will override the original installed value. Endpoint Override DVMs can be edited using the Oracle SOA Composer.

The Itron OpenWay endpoint override DVM (D8–EndpointOverrides.dvm) uses a specific set of keys, each used for one or more commands. The table below lists the DVM keys available for the Itron OpenWay adapter and the command used with each:

DVM Key	Commands
DataService390	Scheduled Read (Scalar) Scheduled Read (Interval) On-Demand Read (Scalar) On-Demand Read (Interval)
ProvisioningService370	Device Commissioning
ProvisioningService390	Device Decommissioning
ControlService370	Device Status Check (if using v3.70) Remote Connect (if using v3.70)

DVM Key	Commands
	Remote Disconnect (if using v3.70)
ControlService390	Remote Connect (if using v3.90)
	Remote Disconnect (if using v3.90)
DiagnosticService390	Device Status Check (if using v3.90)
ProcessCallout	User Exit Functions

NOTE:
The numbers in the keys above (370 and 390) designate the version of the Itron OpenWay head-end system.

Only a single version of each key should be defined, based on the version of the Itron OpenWay head-end system

To define an override Endpoint URI for the Itron OpenWay adapter, use the following procedure:

1. Open the SOA Composer for your BPEL configuration.

   The URL for the SOA Composer is http://server:port/soa/composer.

2. Select the D8–EndpointOverrides.dvm in the left panel.

   - In Fusion Middleware v12.2.1, this is located under **Shared** in the **Deployment View**, or under **Domain Value Maps** in the **Types View**.
   - In Fusion Middleware v12.2.2, this is located under **Metadata** in the **Deployment View**.

   The DVM will open in the right panel. The **Description** field lists the available keys for the DVM (only the first key is shown, but you can scroll through the contents to view the list. The panel also displays a list of previously defined keys.

3. Click the **Create Session** button (above the left panel) to begin an editing session.

4. Click the **Add Domain Values** icon (“+”) to add a new key.

   The **Add Domain Values** dialog opens.

5. Enter the appropriate values in the **Add Domain Values** dialog as follows:

   - **key**: The DVM key for the Endpoint URI you wish to define (see the table above).
   - **EndpointURI**: The override Endpoint URI.

6. Click **OK**.

   The new DVM value will appear in the list of keys.

7. Click the **Save** icon to save the DVM values.

8. Click the **Publish** button to activate all the changes in the editing session.

   Enter an optional note for the session in the **Publish Session** dialog.

   Changes take effect immediately upon publishing the session.

   Click **Discard** to discard your changes.

   Click **Exit** to exit your current session. Note that your session will still be open if you exit. Use **Discard** to end your session without making changes.

Configuring Itron OpenWay Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Itron OpenWay.

This section outlines some of the extendable lookups that must be configured for use with the Itron OpenWay adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Application Group Lookup

The Itron OpenWay head-end system allows users to define addressable groups of meters based on criteria such as a geographic identifier or zip code. These application groups can be represented in Oracle Utilities Smart Grid Gateway by using the Itron OpenWay Application Group Lookup. The groups defined for this lookup are used with Scheduled Read commands. Each value defined for the Itron OpenWay Application Group extendable lookup should include the following:

- **Application Group:** The application group name as defined in the Itron OpenWay head-end system.
- **Description:** A description of the application group.

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system. Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name**: The event name used by the head-end system
- **Description**: A description of the device event
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Standard Event Name**: The standard event name for device events of this type, from the “Standard Event Name” extendable lookup.

Disconnect Switch State Lookup

Some utilities may require verification of the load on a meter to ensure that it is safe to connect the meter. One method of doing this is to check the disconnect switch on the meter to see if it is connected or armed. To use this method, the Itron Openway Disconnect Switch State extendable lookup should be configured to list the possible states of the meter disconnect switch. Each value defined for the Itron OpenWay Interval Disconnect Switch State extendable lookup should include the following:

- **Switch State**: The Itron OpenWay disconnect switch state code
- **Description**: A description of the disconnect switch state code.

UOM Code to Standard UOM Mapping

Usage received from a utility’s head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The UOM Code to Standard UOM Mapping extendable lookup is used for this purpose. Each value defined for the UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-End Unit of Measure**: The unit of measure code used by the Itron OpenWay application
- **Description**: A description of the unit of measure code.
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Unit of Measure**: The unit of measure defined in the system.
- **Time of Use**: An optional time of use period, defined in the system, used to further distinguish the unit of measure.
- **Service Quantity Identifier**: An optional service quantity identifier, defined in the system, used to further distinguish the unit of measure.
- **Unit of Measure Magnitude Conversion**:

- **Magnitude Multiplier**: An optional; multiplier used to convert raw values received from the head-end system to values appropriate for use with the system. For example, if a reading or interval data is received in Wh, a multipler of ".001" would convert Wh to kWh. If not provided or left blank, no conversion is performed.

Interval Status Code to Condition Mapping

Interval usage received from the Itron OpenWay application can include Itron OpenWay interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The Itron OpenWay Interval Status Code to Condition Mapping extendable lookup is used to determine how to map Itron OpenWay interval status codes to standard status codes when receiving usage from the Itron OpenWay application. Each value defined for the Itron OpenWay Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status**: The Itron OpenWay interval status code
- **Description**: A description of the interval status code.
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Condition**: The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.

Other Extendable Lookups

Business Object Name	Description
D8-DvcStatusMappingLookup	Itron - Device Status Lookup
D8-FailureReasonLookup	Itron - Failure Reason Lookup
D8-ItronVersions	Itron - Versions Lookup
D8-JobStatusLookup	Itron - Job Status
D8-ResultLookup	Itron - Result Lookup
D8-RptSelfReadOptionLookup	Itron - Report Self Read Option Lookup

Using the Itron OpenWay Test Harness

Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay includes a test harness that can be configured to simulate a general head-end system for testing the two-way commands. The test harness includes a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. See Using Smart Grid Gateway Test Harnesses for more information.

Landis+Gyr

The Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr supports communication with the Landis+Gyr Gridstream Command Center, including measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, and on-demand read. The following table describes the attributes of the adapter:

Attribute	Details
Currently Supported Version	Gridstream Command Center 7.5
Smart Meter Command Format	MultiSpeak v3.1
Bulk Usage/Event Data Format	California Metering Exchange Protocol (CMEP)

Attribute	Details
Market(s)	North America, portions of Asia Pacific, Sweden, Australia, New Zealand, and Latin America.
Commodities Supported	Electricity, Gas, Water
Architecture	PLC and RF

Landis+Gyr Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, and Oracle Utilities Application Framework (OUAF) objects that are supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data and Device Event Loading

The initial measurement data load and device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway (SGG) service format. Payloads contain measurements and meter events in some head-end specific format. OSB then places each service call into a Java Message Service (JMS) queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel. A service then creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format.

Initial Measurements

The usage data exported from the AMI head-end system as a file in Landis+Gyr format is loaded into Oracle Utilities as initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. **SGG-D3-USAGE-BASE** - contains components responsible for “actual” processing of incoming data. It should not be modified during configuration. This can be upgraded without affecting the customization and environment settings added to SGG-D3-USAGE-CM.
2. **SGG-D3-USAGE-CM** allows for customization and simplifies future upgrades.

When importing non-interval usage data, separate initial measurements can be created for difference measurement types. For instance, if the data includes Power Factor or Volt data, separate initial measurements are created for each of these. See Non-Interval 'Plain' XML to IMD Mapping for more information about specific units of measure that trigger the creation of separate initial measurements.

The runtime configuration settings for the SGG-D3-USAGE-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRawIMD	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	

Element	Description	Valid Values
destinationRootElementScalar	Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval.	
modifyResultXMLInput	Specifies the name of an XQuery document (without the "xq" extension) used to map additional fields from the "plain" XML format to the result XML format sent as initial measurement data. See Mapping Additional Fields for more information.	
dateTimeInUTC	Indicates whether the Landis+Gyr system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device.	true false
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterUsage	Determines if usage should be filtered.	true false

Publishing Initial Measurement Data

The SGG adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Initial Measurement Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D3-USAGE-CM OSB project are used in publishing measurement data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Initial Measurement Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Initial measurement data is published in the “native” initial measurement data format (the format of the initial measurement seeder business object). This format includes normalized unit of measure and condition codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

> **NOTE:** Initial measurement data published through this feature is published prior to validation, estimation, and editing (VEE) processing. In addition, filtering *cannot* be applied to data published through this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Initial Measurement Data

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **Landis +Gyr UOM Code to Standard UOM Mapping** extendable lookup (D3-HeadendUOMLookup) are passed into the system for processing.

NOTE: Filtering of scalar initial measurement data is not supported in the Landis+Gyr adapter.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service for the project. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Prioritized Initial Measurement Processing

The SGG adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to “Real Time” (D1RT). In addition, initial measurements received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Device Events

The device event data exported from the head-end system as a file in Landis+Gyr format is loaded into Oracle Utilities as a Device Event. One of your configuration tasks is to customize the device events processing. The required functionality is delivered in the base product as two OSB projects:

1. **SGG-D3-EVENT-BASE** containing components responsible for "actual" processing of incoming data. It can be upgraded in future without affecting the customization and environment settings that done in SGG-D3-EVENT-CM project.
2. **SGG-D3-EVENT-CM** allows the customization and simplifies the future upgrades.

The runtime configuration settings for the SGG-D3-EVENT-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element. The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the device event data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElement	Holds the name of inbound web service for the device event seeder.	
modifyResultXMLInput	Specifies the name of an XQuery document (without the "xq" extension) used to map additional fields from the "plain" XML format to the result XML format sent as device event data. See Mapping Additional Fields for more information.	
dateTimeInUTC	Indicates whether the Landis+Gyr system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device.	true false
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false

Publishing Events

SGG can be configured to publish device events for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Device Event Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D3-EVENT-CM OSB project are used in publishing device events data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Device Event Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Device event data is published in the "native" device event data format (the format of the device event seeder business object). This format includes normalized device event codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: In addition, filtering can NOT be applied to device events published via this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Events

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to "true" as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **Landis+Gyr Device Event Mapping** extendable lookup (D3-DeviceEventMappingLookup) are passed into the system for processing.

Subscribing to Real-Time Device Events

The Landis+Gyr Command Center provides the ability to "subscribe" to device events from within their online interface. This is done by creating a subscriber in Command Center with an associated callback URL as well as a list of events types that subscriber is interested to receive. When an event that is subscribed to occurs it is sent to the callback URL in real time.

Within L+G Events are configured with one of three "alarm" settings. These settings determine how often the events will be sent to subscribers:

1. Alarm: immediately delivered from the meter
2. Advisory: sent based upon a delivery schedule
3. Log Only: sent only upon request (not applicable for our implementation real time event processing)

Command Center will communicate the events using a CIM format that describes the message as a noun/verb combination. The details of the event itself will be contained within a "payload" element of the standard structure. The payload will be formatted using the EndDeviceEvent message structure. This message identifies device events using a CIM 4-part category number. These numbers are four period separated numbers that will describe the type of device and the event. For example: 3.33.1.257 is for "Tamper attempt suspected".

- Segment 1: End Device event domain code (e.g. 3. meter/10. collector/11. router/12. HAN device)
- Segment 2: End Device Event Domain Part Codes (e.g. 1. Access/2. Battery)
- Segment 3: End Device Event Type Codes (e.g. 1. Alarm/2. Alarm Mgt)
- Segment 4: End Device Event Index (e.g. 1. Abort/2. Access Attempt)

Refer to the Landis+Gyr documentation for details about the CIM Category Numbers. CIM Category Numbers must be mapped to standard device event names using the Landis+Gyr Device Event Mapping extendable lookup.

SGG receives these messages through a BPEL composite that saves the incoming request as a file to be picked up by OSB.

The **AMIEventSubscriber** composite is responsible for receiving the event messages based on subscriptions defined in the L+G Command Center. The callback URL configured for the subscription in the Command Center should point to this BPEL composite.

The following OSB projects parse individual device events from the message and perform the validation and mapping of the information to the Device Event Seeder Format.

1. **SGG-D3-CIM-EVENT-BASE** contains components responsible for "actual" processing of incoming data. It can be upgraded in future without affecting the customization and environment settings that done in SGG-D3-CIM-EVENT-CM project.
2. **SGG-D3-CIM-EVENT-CM** allows the customization and simplifies the future upgrades.

The runtime configuration settings for the SGG-D3-CIM-EVENT-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the device event data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false

Element	Description	Valid Values
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElement	Holds the name of inbound web service for the device event seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false

Processing statistics are gathered for any real time events that are received (even if there is just one event in the message) in the same manner as device events received via the flat-file interface.

Prioritized Device Event Processing

The SGG adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Configuring Payload Processing Notifications

By default, Payload Statistics, Payload Error Notification, and Payload Summary notification activities are sent to the Notification Queue within Oracle Service Bus. Smart Grid Gateway adapters can be configured to restrict sending these activities by defining optional JCA transport dynamic properties in the appropriate OSB project. The following optional JCA transport dynamic properties can used to control if these notification activities are sent:

- **SGG_SEND_STATISTICS**: Controls sending Payload Statistic activities (true: send, false: do not send)
- **SGG_SEND_ERROR**: Controls sending Payload Error Notification activities (true: send, false: do not send)
- **SGG_SEND_SUMMARY**: Controls sending Payload Summary activities (true: send, false: do not send)

When these properties are set to false, the corresponding payload notification will not be sent to the JMS Queue. Note these properties are not populated by default, and default to true if omitted.

To create these properties:

1. Navigate to the SGG-D3-CIM-EVENT-CM project in the Oracle Service Bus Console.
2. Expand the **Proxy Services** folder.
3. Select the **InboundProxyService** proxy service (not the Pipeline or WSDL).
4. Click **Create** to create a new session.
5. Select **Transport Details**.
6. Click the plus sign in the **Dyanmic EndPoint Properties** section. An empty row will appear.
7. Enter the property you wish to define in the **Property** column.
8. Enter "false" in the **value** colulmn.
9. Repeat steps 6-8 for each property you wish to define.
10. Click **Activate** to save and activate your changes.

NOTE: These properties are only used if the SGG_EXT_ID_PATTERN is not present or the regular expression captures nothing. The SGG_EXT_ID_PATTERN will override these properties when the regular expression captures group data.

Base Package Business Objects

The Landis+Gyr adapter base package includes the following initial measurement business objects:

Business Object Name	Description
D3-InitialLoadIMDInterval	Landis+Gyr Initial Load IMD - Interval Used when loading Landis+Gyr interval measurements into the system for the first time.
D3-InitialLoadIMDScalar	Landis+Gyr Initial Load IMD - Scalar

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, an on-demand read, or another type of request that the head-end system supports. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related web service from the head-end system. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

Communication Flows

The table below lists the communications created for each Landis+Gyr command:

Command	Outbound Communication	Inbound Communication	Completion Event
Remote Connect (This command has sub-commands)	Initiate MR by Mtr Num Initiate Connect Disconnect	Reading Changed Notification Connect Disconnect State Changed Notification	Connect Device Create IMD Completion Event
Remote Disconnect (This command has sub-commands)	Initiate Connect Disconnect Initiate MR by Mtr Num	Connect Disconnect State Changed Notification Reading Changed Notification	Disconnect Device Create IMD Completion Event
Device Commissioning (Registration)	L+G Add Meter to Inventory		Device Commissioning
Device Commissioning (Installation)	L+G Meter Exchange Notification		Device Commissioning
Device Decommissioning	Meter Remove Notification		Device Decommissioning
Device Deregistration	L+G Meter Retire Notification		Device Deregistration
On-Demand Read (Scalar)	Initiate MR by Mtr Num	Reading Changed Notification	Create IMD Completion Event
On-Demand Read (Scalar) - CIM	CIM Meter On Demand Read (Scalar)	CIM Meter On Demand Read Response	Create IMD Completion Event
On-Demand Read (Interval)	Initiate MR by Mtr Num	Reading Changed Notification	Create IMD Completion Event
On-Demand Read (Interval) - CIM	CIM Meter On Demand Read (Interval)	CIM Meter On Demand Read Response	Create IMD Completion Event
Device Status Check	CIM Ping	CIM Ping Response	
Demand Reset	Schedule Demand Reset (Multispeak)	Schedule Demand Reset Response (Multispeak)	Create IMD Completion Event

Device Registration Commission Commands

Landis+Gyr device commission commands can be used to “register” the device and notify the L+G head-end system that meters have been added to inventory. Device commission commands of this type have the “Registration-Only Mode” flag set to “Yes”. An Enter algorithm on the “Commission Ready” state evaluates the Registration-Only Mode of the command and if set to “Yes”, the command skips the default “Waiting for Measurement” state and is transitioned to the “Execute Completion Event” state, and an activity log entry is created.

Only the device registration request is sent to the head-end system for device commission commands of this type.

Device registration commands are typically created when new devices are added to inventory in an asset management system such as Oracle Utilities Operational Device Management.

Device Installation Commission Commands

Landis+Gyr device commission commands can be used to notify the L+G head-end system that meters have been installed or exchanged. An Enter algorithm on the “Commission Ready” state evaluates the “Is Installation Check Unnecessary” flag of the command and if set to “False”, the algorithm creates an “L+G Meter Exchange Notification” outbound communication and sends an installation notification to the head-end system.

Device Deregistration Commands

The Landis+Gyr adapter supports the Device Deregistration command (based on the D1-DeviceDeregistration business object). This command sends a communication that deregisters the device in the head-end system, and is most often used when retiring a device. The specific message sent is defined for the Device Deregistration processing role for the L+G head-end system service provider.

Device deegistration commands are typically created when devices are retired in an asset management system such as Oracle Utilities Operational Device Management.

Device Communication Base Package Business Objects

The Landis+Gyr Adapter base package includes the following communication business objects:

Business Object Name	Description	Protocol
D3-AddMeterToInventoryMultiSp	L+G Add Meter to Inventory (MultiSpeak)	MultiSpeak
D3-CIMGetLPData	CIM Meter On Demand Read (Interval)	CIM
D3-CIMMeterOnDemandRead	CIM Meter On Demand Read (Scalar)	CIM
D3-CIMMeterReadingResponse	CIM Meter On Demand Read Response	CIM
D3-CIMPing	CIM Ping	CIM
D3-CIMPingResponse	CIM Ping Response	CIM
D3-ConnectDisconStateChgNtf	Connect Disconnect State Changed Notification	MultiSpeak
D3-InitiateConnectDisconnect	Initiate Connect Disconnect	MultiSpeak
D3-InitiateMRByMtrNbr	Initiate Meter Read By Meter (MultiSpeak)	MultiSpeak
D3-MeterAddNotificationMultiSp	Meter Add Notification (MultiSpeak)	MultiSpeak
D3-MeterExNotificationMultiSp	L+G Meter Exchange Notification	MultiSpeak
D3-MeterRetireNotification	L+G Meter Retire Notification	MultiSpeak
D3-MtrRmvNotifMultiSpeak	Meter Remove Notification (MultiSpeak)	MultiSpeak
D3-ReadingChgNotification	Reading Changed Notification	MultiSpeak
D3-ScheduleDemandReset	Schedule Demand Reset (Multispeak)	MultiSpeak
D3-ScheduleDemandResetResponse	Schedule Demand Reset Response (Multispeak)	MultiSpeak

Landis+Gyr Event Data Mapping

The Landis+Gyr event file format maps as follows into the business object, D1-DeviceEventMappingLookup:

Landis+Gyr Flat File Field	Device Event Seeder BO Element	Comments
Transaction ID (from Header record)	External Source Identifier	This is the file name.
Device Identifier	External Device Identifier	
Event Name	External Event Name	
Event Creation Date/Time	Event Date/Time	
Device Type	External Device Type	This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form): Meter Collector Router
Service Location ID	External Service Location ID	
Communication Module Serial Number	External Communication Module Identifier	
Event Category ID	External Event Category	
Event Severity	External Event Severity	Valid values include (although the element itself is free-form): Alert Information
Status Value	External Status Value	This represents additional information that relates to the event itself.
Status Date/Time	External Status Date/Time	The date & time at which the additional information referenced above had occurred.

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch or Real-time)
- The corresponding message senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgment and response messages are sent and received validating that commands have been transmitted. These notifications are based on the following outbound message types.

Outbound Message Type	Description
D3-ADDMTRINV	Add Meter to Inventory

Outbound Message Type	Description
D3-COMMS	Commission Device
D3-CONNECT	Connect Device
D3-DECOMMS	Decommission
D3-DEMRESET	Demand Reset
D3-DERDEV	Deregister Device
D3-DISCONNEC	Disconnect Device
D3-DVCSTSCHK	Device Status Check
D3-INITMRN	Initiate Meter Read by Meter Number
D3-INITMTR	Initiate Meter Read by Meter Number
D3-MTRADDNOT	Meter Add Notification Outbound Message Type
D3-MTREX	Meter Exchange Notification OB MSG
D3-MTRRMV	Meter Remove Notification

Inbound / Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The Oracle Utilities Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway adapter for Landis+Gyr includes the following inbound web services:

Inbound Web Service	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Used for upload of device events. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-InitialLoadIMD	Used for initial measurement upload. The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.
D3-CIMMeterReadingsResponse	CIM Meter On Demand Read Response Retrieve response from CIM On Demand Read command
D3-CIMPingResponse	CIM Ping Response Retrieve response from CIM Device Status Check command

Inbound Web Service	Description
D3-ConDisconStChgNotification	Initiate Connect Disconnect response. Retrieve response from the Initiate Connect Disconnect command.
D3-ReadingChangedNotification	Reading Changed Notification Notification that a Landis+Gyr device reading has changed.
D3-ScheduleDemandResetResponse	Scheduled Demand Reset Response Retrieve response from Demand Reset command

Message Senders

Message senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of message senders you need to create is based on the types of messages the system is designed to accept.

The Oracle Utilities Smart Grid Gateway adapter for Landis+Gyr includes the following message senders:

Message Sender	Description
D3-Comms	Commission Device
D3-Connect	Connect Device
D3-Decomm	Decommission Device
D3-Decomms	Decommissioning Sender
D3-DemReset	Demand Reset
D3-DerDevice	Deregister Device
D3-Disconnec	Disconnect Device
D3-InitMTR	Initiate Meter Read by Meter Number Outbound Message
D3-MTREXMS	Meter Exchange Notification Message Sender
D3-RTSender	Real Time Sender
D3-RTSnd	Real-time Sender (Landis+Gyr)
D3-SDemReset	SG Demand Reset

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to MultiSpeak 3.0 format, invoking process callouts and invoking the remote endpoint to trigger the device events.

OnDemandRead Composite Process: Invokes the remote endpoint to trigger the on-demand read event. An asynchronous reply responds to the OUAF layer when the reading arrives.

ConnectDisconnect Composite Process: Invokes the remote endpoint to trigger the connect/disconnect event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

CommissionDecommission Composite Process: Invokes the remote endpoint to trigger the commission or decommission event. After the synchronous call completes, one of the following second business callout services is invoked to determine if the related “received” or “completed” callout should be executed:

- isExecutingCommissionReceivedCallout
- isExecutingCommissionCompletedCallout

- isExecutingDecommissionReceivedCallout
- isExecutingDecommissionCompletedCallout
- isExecutingAddMeterToInventoryReceivedCallout
- isExecutingAddMeterToInventoryCompletedCallout
- isExecutingMeterExchangeNotificationReceivedCallout
- isExecutingMeterExchangeNotificationCompletedCallout

CIMOnDemandRead Composite Process: Invokes the remote endpoint to trigger the CIM on-demand read event. An asynchronous reply responds to the OUAF layer when the reading arrives.

CIMDeviceStatusCheck Composite Process: This process is similar to CIM OndemandRead BPEL process. SGG uses the result of On Demand Read command to identify the status of the meter. If reads are successfully returned, then meter is running healthy otherwise it is considered as meter inactive/dead.

DemandReset Composite Process: Invokes the remote endpoint to trigger the demand reset event. An asynchronous reply responds to the OUAF layer when the reading arrives.

LGProcessCallout Composite: This business callout provides a point at which customers and implementers can incorporate custom business logic and transformations. This composite includes the WSDLs and processing logic for all of the MultiSpeak processes. The default implementation of each method is a direct return of the input.

Web Services

These web services are all defined in the Landis+Gyr head end system. The WSDLs were added to a Meta Data Storage (MDS) layer in OUAF and all references to the WSDL point to this MDS location.

Web Service	Related BPEL Process	Description
MR_CB	OnDemandRead CommissionDecommission DemandReset	This web service is defined by the Landis+Gyr head end system's implementation of MR_Server. The WSDL defines the interface for requesting a meter reading from the head end system. The actual definition can be obtained from L&G or downloaded from multispeak.org. Build 3.0aa is appropriate if obtained from MultiSpeak. Default endpoint must be changed in configuration: http://demo.turtletech.com/Multispeak/webapi/MR_CB.asmx
CD_CB	ConnectDisconnect	This web service is defined by the Landis+Gyr implementation of CB_CD. The WSDL defines the interface for requesting a meter's connection or disconnection on the head end system. This web service defines the interface for reporting a connection or disconnection by the head end system. This web service is only invoked by the head end system; not OUAF. Only the CDStateChangedNotification web method is implemented in the composite. Default endpoint must be changed in configuration: http://demo.turtletech.com/Multispeak/webapi/CD_CB.asmx
CIMService	CIMOnDemandRead CIMDeviceStatusCheck	This web service is defined by the L+G head end's implementation of AMIRequest Server.

Web Service	Related BPEL Process	Description
		The WSDL defines the interface for requesting a meter reading from the head end system. The actual definition should be obtained from L&G or downloaded from L&G SDK for CIM 2.0.
LGProcessCallout	OnDemandRead ConnectDisconnect CommissionDecommission	Imported from **LGProcessCallout Composite** Default endpoint must be changed in configuration: http://127.0.0.1:8000/soa-infra/services/default/LGProcessCallout/LGProcessCallout

Landis+Gyr Command Center Web Services

The following table describes the Land+Gyr Command Center web services and operations used for the Oracle Utilities Smart Grid Gateway command messaging:

Smart Grid Gateway Command	AMI Adapter Business Objects	Landis+Gyr Web Services	Landis+Gyr Operations
Device Commissioning	D3-MeterAddNotificationMultiSp	MR	MeterAddNotification
Device Decommissioning	D3-MtrRmvNotifMultiSpeak	MR	MeterRemoveNotification
Remote Connect/ Remote Disconnect	D3-InitiateConnectDisconnect	CD	InitiateConnectDisconnect
On-Demand Read	D3-InitiateMRByMtrNbr	MR	InitiateMeterReadByMeterNumber
On-Demand Read (CIM)	D3-CIMGetLPData D3-CIMMeterOnDemandRead	CIMService	ScheduleDemandRead
Demand Reset	D3-ScheduleDemandReset	MR	CIM

Configuring a Landis+Gyr Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr to communicate with the Landis+Gyr Command Center software.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the L+G Command Center in response to a command request.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Device Event Seeder
D1-InitialLoadIMD	IMD Seeder
D1-PayloadErrorNotif	Payload Error Notification
D1-PayloadStatistics	Payload Statistics

Inbound Web Service Name	Description
D1-PayloadSummary	Payload Summary
D3-ConDisconStChgNotification	Initiate Connect Disconnect Response
D3-CIMMeterReadingsResponse*	CIM Meter On Demand Read Response
D3-CIMPingResponse	CIM Ping Respose
D3-ReadingChangedNotification*	Reading Changed Notification
D3-ScheduleDemandResetResponse	Scheduled Demand Reset Response

*The Landis+Gyr adapter supports both MultiSpeak and CIM On Demand Read commands. You only need to configure the inbound service for the protocol you wish to use.

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object
- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

Message Senders

Message senders define the details of how messages are sent to an external system, such as messages containing device command requests.

The following message senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders.

Message Sender	Description
D3-Comms	Commission Device
D3-Connect	Connect Device
D3-Decomm	Decommission Device
D3-InitMTR	Initiate Meter Read by Meter Number Outbound Message
D3-RTSender	Real Time Sender
D3-RTSnd	Real-time Sender (Landis+Gyr)
D3_CIMODR	L+G CIM On Demand Read
D3_CIMPING	L+G CIM Ping
D3_SCHDEMRES	L+G Schedule Demand Reset

Note: The following apply to all of the above message senders:

Main Tab:

- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding**: UTF-8 message encoding

Context Tabsysuser00:

- **HTTP Header**: SOAPAction: http://xmlns.oracle.com/ouaf/multispeak_3.0/**<OPERATION>**
- **HTTP Login User**: **<USER_ID>**
- **HTTP Login Password**: **<PASSWORD>**
- **HTTP Method**: POST
- **HTTP URL 1**: http://**<EM_SERVER>**:**<EM_SERVER_PORT>**/soa-infra/services/D3/**<SERVICE>**/**<SERVICE>**

  where:

  - **<OPERATION>**: the operation performed by the message sender (see Operation column in the table above)
  - **<USER_ID>**: the user ID used to log into WebLogic Enterprise Manager
  - **<PASSWORD>**: the password used to log into WebLogic Enterprise Manager
  - **<EM_SERVER_IP>**: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
  - **<EM_SERVER_PORT>**: the port where the WebLogic Enterprise Manager is installed
  - **<SERVICE>**: the service invoked by the message sender (see Service column in the table above)

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
D3-ADDMTRINV	Add Meter to Inventory
D3-COMMS	Commission Device
D3-CONNECT	Connect Device
D3-DECOMMS	Decommission
D3-DEMRESET	Demand Reset
D3-DERDEV	Deregister Device
D3-DISCONNEC	Disconnect Device
D3-DVCSTSCHK	Device Status Check
D3-INITMRN	Initiate Meter Read by Meter Number
D3-INITMTR	Initiate Meter Read by Meter Number
D3-MTRADDNOT	Meter Add Notification Outbound Message Type
D3-MTREX	Meter Exchange Notification OB MSG
D3-MTRRMV	Meter Remove Notification

Note: The following apply to all of the above outbound message types:

- **Business Object**: D1-OutboundMessage (Outbound Message)
- **Priority**: Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the L+G Command Center must be present in your system. If this is not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System - Landis+Gyr:

- **External System**: LG
- **Description**: Landis+Gyr
- **Outbound Message Types**::

Outbound Message Type	Description	Message Sender
D3-ADDMTRINV	Add Meter to Inventory	Message sender associated with the Add Meter to Inventory Outbound Message Type
D3-COMMS	Commission Device	Message sender associated with the Commission Device Outbound Message Type
D3-CONNECT	Connect Device	Message sender associated with the Connect Device Outbound Message Type
D3-DECOMMS	Decommission	Message sender associated with the Decommission Device Outbound Message
D3-DEMRESET	Demand Reset	Message sender associated with the Demand Reset Outbound Message Type
D3-DERDEV	Deregister Device	Message sender associated with the Deregister Device Outbound Message Type
D3-DISCONNEC	Disconnect Device	Message sender associated with the Disconnect Device Outbound Message Type
D3-DVCSTSCHK	Device Status Check	Message sender associated with the Device Status Check Outbound Message Type
D3-INITMRN	Initiate Meter Read by Meter Number	Message sender associated with the Initiate Meter Read By Meter Number Outbound Message Type
D3-INITMTR	Initiate Meter Read by Meter Number	Message sender associated with the Initiate Meter Read By Meter Number Outbound Message Type
D3-MTRADDNOT	Meter Add Notification Outbound Message Type	Message sender associated with the Meter Add Notification Outbound Message Type
D3-MTREX	Meter Exchange Notification OB MSG	Message sender associated with the Meter Exchange Notification Outbound Message Type
D3-MTRRMV	Meter Remove Notification	Message sender associated with the Meter Remove Notification Outbound Message Type

Note: The following apply to the above outbound message types*:

- **Processing Method**: Real-time
- **Message XSL**: D3-Request xsl (*use D3-DRRequest.xsl for the Demand Reset command)
- **Response XSL**: D3-Response.xsl (*use D3-DRResponse.xsl for the Demand Reset command)

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the L+G Command Center must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about creating service providers.

Service Provider - Landis+Gyr:

- **Service Provider**: LG
- **Description**: Landis+Gyr
- **External Reference ID**: L+G
- **External System**: Landis+Gyr
- **Out Name/ID in Their System**:
- **AMI Device ID Type**: Internal Meter Number
- **AMI Measuring Component ID Type**: Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to a given head-end system.

The following types of processing methods must be configured for the L+G service provider. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head-end system.

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.

UOM Translation

UOM mapping processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the L+G service provider, based on the requirements of each implementation using the “How to Create OB COMM/Send OB Message” processing method business object (D1–HowToCreateActivityOBComm).

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Registration	D3-AddMeterToInventoryMultiSp	Add Meter to Inventory

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Installation	D3-MeterExNotificationMultiSp	Meter Exchange Notification
Device Decommission	Device Removal	D3-MtrRmvNotifMultiSpeak	Decommission Meter Remove Notification
Device Deregistration	Device Deregistration	D3-MeterRetireNotification	Deregister Device
On-Demand Read (Scalar), Multispeak and CIM	On-Demand Read (Scalar)	MS: D3-InitiateMRByMtrNbr CIM: D3-CIMMeterOnDemandRead	MS: Initiate Meter Read by Meter Number CIM: CIM On Demand Read
On-Demand Read (Interval)	On-Demand Read (Interval)	D3-CIMGetLPData	CIM On Demand Read
Demand Reset	Demand Reset	D3-ScheduleDemandReset	Demand Reset
Device Status Check	Device Status Check	D3-CIMPing	Device Status Check
Remote Connect	Remote Connect	D3-InitiateConnectDisconnect	Connect Device
Remote Disconnect	Remote Disconnect	D3-InitiateConnectDisconnect	Disconnect Device

Configuring Endpoint URIs

Part of the configuration process is configuring your BPEL composites to work with your head-end system by defining the appropriate Endpoint URIs for each of the commands. The default approach to defining Endpoints URIs is redeployment or reinstallation or the BPEL composites. For example, by default, changing an adapter from using the test harness to a production environment using the actual head-end system requires editing the appropriate installation menu options and redeploying the BPEL composites. The Endpoints URIs defined during installation and deployment for each adapter are listed in the **Smart Grid Gateway Installation and Configuration Worksheets** section of the *Oracle Utilities Smart Grid Gateway Installation Guide*.

You can also use an “Endpoint Override” Domain Value Map (DVM) to override Endpoint URIs defined during deployment and installation. This DVM allows defining specific keys that provide an alternate URL that will override the original installed value. Endpoint Override DVMs can be edited using the Oracle SOA Composer.

The Landis+Gyr endpoint override DVM (D3–EndpointOverrides.dvm) uses a specific set of keys, each used for one or more commands. The table below lists the DVM keys available for the Landis+Gyr adapter and the command used with each:

DVM Key	Commands
CIMService	On-Demand Read (Scalar) - CIM On-Demand Read (Interval) - CIM Device Status Check
CD_CB	Remote Connect Remote Disconnect
Metering	Device Commissioning (Registration)
MR_CB	Device Commissioning Device Decommissioning Device Commissioning (Installation) Device Deregistration On-Demand Read (Scalar) On-Demand Read (Interval)
LGProcessCallout	User Exit Functions

To define an override Endpoint URI for the Landis+Gyr adapter, use the following procedure:

1. Open the SOA Composer for your BPEL configuration.

   The URL for the SOA Composer is http://server:port/soa/composer.

2. Select the D3–EndpointOverrides.dvm in the left panel.

   - In Fusion Middleware v12.2.1, this is located under **Shared** in the **Deployment View**, or under **Domain Value Maps** in the **Types View**.
   - In Fusion Middleware v12.2.2, this is located under **Metadata** in the **Deployment View**.

   The DVM will open in the right panel. The **Description** field lists the available keys for the DVM (only the first key is shown, but you can scroll through the contents to view the list). The panel also displays a list of previously defined keys.

3. Click the **Create Session** button (above the left panel) to begin an editing session.

4. Click the **Add Domain Values** icon ("+") to add a new key.

   The **Add Domain Values** dialog opens.

5. Enter the appropriate values in the **Add Domain Values** dialog as follows:

   - **key**: The DVM key for the Endpoint URI you wish to define (see the table above).
   - **EndpointURI**: The override Endpoint URI.

6. Click **OK**.

   The new DVM value will appear in the list of keys.

7. Click the **Save** icon to save the DVM values.

8. Click the **Publish** button to activate all the changes in the editing session.

   Enter an optional note for the session in the **Publish Session** dialog.

   Changes take effect immediately upon publishing the session.

   Click **Discard** to discard your changes.

   Click **Exit** to exit your current session. Note that your session will still be open if you exit. Use **Discard** to end your session without making changes.

Configuring Landis+Gyr Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Landis+Gyr.

This section outlines some of the extendable lookups that must be configured for use with the Landis+Gyr adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

CIM Response Status Extendable Lookup

The CIM Response Status extendable lookup is used to map descriptions to response status codes received from the L+G Command Center. Each value defined for the CIM Response Status extendable lookup should include the following:

- **Response Status**: The CIM status code for the response status
- **Description**: A description of the response status
- **Status**: The status of the lookup value (can be Active or Inactive)

CIM Data Source Extendable Lookup

The CIM Data Source extendable lookup is used to map descriptions to data sources defined in the L+G Command Center. Each value defined for the CIM Data Source extendable lookup should include the following:

- **Data Source**: The CIM code for the data source
- **Description**: A description of the data source
- **Status** : The status of the lookup value (can be Active or Inactive)

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system. Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name**: The event name used by the head-end system
- **Description**: A description of the device event
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Standard Event Name**: The standard event name for device events of this type, from the “Standard Event Name” extendable lookup.

UOM Code to Standard UOM Mapping

Usage received from a utility’s head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The UOM Code to Standard UOM Mapping extendable lookup is used for this purpose. Each value defined for the UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-end UOM**: The unit of measure code used by the head-end system.
- **Unit of Measure**: The unit of measure defined in the system.
- **Description**: A description of the unit of measure code.

Interval Status Code to Condition Mapping

Interval usage received from the Landis+Gyr Command Center can include interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The Landis+Gyr Interval Status Code to Condition Mapping extendable lookup is used for this purpose. Each value defined for the Landis+Gyr Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status**: The interval status code
- **Condition**: The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.
- **Description**: A description of the interval status code.

Using the Landis+Gyr Test Harness

Oracle Utilities Smart Grid Gateway Adapter for Landis+Gyr (LG) includes a test harness that can be configured to simulate the Landis+Gyr Gridstream Command Center head-end system for testing the two-way commands. The test harness is Multispeak 3.0 standard compliant and includes a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. See Using Smart Grid Gateway Test Harnesses for more information.

Landis+Gyr Interval Data Mapping

This section describes how data in Landis+Gyr import files is mapped to the XML document format created by OSB and sent to Oracle Utilities Smart Grid Gateway.

When Landis+Gyr data is processed, it is initially received in a tilda-separated file format, which is converted into a "plan" XML format before being converted into the "result" XML format which is sent to the IMD Seeder and/or Device Event Seeder inbound services.

Non-Interval Usage with Additional Fields

The following is a sample file of non-interval usage that contains additional fields.

ID~PremiseID~ESIID~Provisioned~Meter~kWh~DateTime~Peak~PeakDateTime~Dmd~TouA~TouB~TouC~TouD~TouE~Volts~PF~Pı

EMED01~SL002~~~96968280~34315.000~08042010120000AM~2.66~03122009063000AM~~34315.000~0.000~0.000~0.000~0.000~~

EMED01~SL001~~~96968285~33693.000~08042010120000AM~2.62~03122009061500AM~~33693.000~0.000~0.000~0.000~0.000~~

This data is mapped to the "plain" XML format.

XML 'Plain' XML Format

The "Plain" XML contain elements to hold the extra fields (highlighted in bold).

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace=" http://xmlns.oracle.com/LandisGyrUsage"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MeterReads">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MeterRead">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Origin"/>
              <xs:element name="ServProvExtRefId"/>
              <xs:element name="RecordType">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:enumeration value="MEPMD01" />
                      <xs:enumeration value="EMED01" />
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
              <xs:element name="RecordVersion" minOccurs="0">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:enumeration value="20080519" />
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
              <xs:element name="TimeStamp" />
              <xs:element name="Premise" minOccurs="0" />
              <xs:element name="ESIID" minOccurs="0" />
              <xs:element name="Provisioned" minOccurs="0" />
              <xs:element name="MeterID" />
              <xs:element name="Purpose" minOccurs="0" />
              <xs:element name="Comodity" minOccurs="0" />
              <xs:element name="Units" minOccurs="0" />
              <xs:element name="CalcConst" minOccurs="0"/>
              <xs:element name="Interval" minOccurs="0"/>
```

```
              <xs:element name="Count" minOccurs="0"/>
              <xs:element name="FirstIntervalDateTime" />
              <xs:element name="Data">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Row" maxOccurs="unbounded" minOccurs="0">
                      <xs:complexType>
                        <xs:attribute name="v" />
                        <xs:attribute name="s" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="kWh" minOccurs="0" />
              <xs:element name="Peak" minOccurs="0" />
              <xs:element name="PeakDateTime" minOccurs="0" />
              <xs:element name="Dmd" minOccurs="0" />
              <xs:element name="TouA" minOccurs="0" />
              <xs:element name="TouB" minOccurs="0" />
              <xs:element name="TouD" minOccurs="0" />
              <xs:element name="TouC" minOccurs="0" />
              <xs:element name="TouE" minOccurs="0" />
              <xs:element name="Volts" minOccurs="0" />
              <xs:element name="PF" minOccurs="0" />
              <xs:element name="ExtraFields" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ExtraField" maxOccurs="255"minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="FieldName" minOccurs="0"/>
                          <xs:element name="FieldValue" minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="RawData"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Non-Interval Usage to 'Plain' XML Mapping

The following table shows the mapping between fields in incoming non-interval data and child elements of MeterReads/ MeterRead element in the "Plain" XML format:

L+G Interval Structure Field	"Plain" XML element
ID	RecordType
Premise ID	Premise
ESIID	ESIID
Provisioned	Provisioned
Meter	MeterID
kWh	kWh
Date/Time of initial Read	FirstIntervalDateTime
Peak	Peak
Peak Date/Time	PeakDateTime
Dmd	Dmd

L+G Interval Structure Field	"Plain" XML element
TouA	TouA
TouB	TouB
TouC	TouC
TouD	TouD
TouE	TouE
Volts	Volts
PF	PF
<extraFieldName1>	ExtraFields/ExtraField/FieldName with value of <extraFieldName1> ExtraFields/ExtraField/FieldValue with value in <extraFieldName1>
<extraFieldName2>	ExtraFields/ExtraField/FieldName with value of <extraFieldName2> ExtraFields/ExtraField/FieldValue with value in <extraFieldName2>
<extraFieldName3>	ExtraFields/ExtraField/FieldName with value of <extraFieldName3> ExtraFields/ExtraField/FieldValue with value in <extraFieldName3>
<extraFieldNameN>	ExtraFields/ExtraField/FieldName with value of <extraFieldNameN> ExtraFields/ExtraField/FieldValue with value in <extraFieldNameN>
RawData	A record content from incoming file.

'Plain' XML to IMD Mapping

The following table outlines how data from the "plain" XML format is mapped to the InitialLoadIMD format when the Landis+Gyr record type is set to "MEPMD01" (interval usage).

"Plain" XML element	InitialLoadIMD element	Note
Record Type	N/A	
RecordVersion	N/A	
Premise ID	N/A	
ESIID	N/A	
Provisioned	N/A	
Meter ID	dvcIdN	as is
Purpose	N/A	
Commodity	N/A	
Units	externalUOM	as is
CalcConst	mcm	as is
Interval	spi	Transform from DDHHMM to SPI
Count	N/A	
FirstIntervalDateTime	stDt	Convert to OUAF date/time format
Data	msrs	Value -> msrs/mL/q
Status -> msrs/mL/sts/stsL/st		
N/A	imdType	'D1IL
Origin	externalId	the origin attribute in "Plain" XML (incoming file name)
N/A	serviceProviderExternalId	L+G
RawData	rawData	Vendor-specific "raw" data

Non-Interval 'Plain' XML to IMD Mapping

The following table outlines how non-interval data in the "plain" XML format is mapped to the InitialLoadIMD format when the Landis+Gyr record type is set to "EMED01" (non-interval usage).

"Plain" XML Element	InitialLoadIMD element	Notes
RecordType	N/A	
Premise	N/A	
ESIID	N/A	
Provisioned	N/A	
MeterID	dvcIdN	as is
kWh	enQty	as is
	uom	KWH
TimeStamp	enDt	Convert to OUAF date/time format This should be mapped to enDt. Relevant only to non-interval data.
Peak	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KW
PeakDateTime	enDt	Convert to OUAF date/time format
Dmd	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KW and 'Dmd' in "<mcIdN>
TouA	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate A' in "<mcIdN>
TouB	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH' and 'kWh Rate B' in "<mcIdN>
TouC	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate C' in "<mcIdN>
TouD	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate D' in "<mcIdN>
TouE	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with KWH and 'kWh Rate E' in "<mcIdN>
Volts	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with Volts
PF	enQty	as is A separate IMD Seeder will be created with <uom> in IMD Seeder populated with PF
N/A	imdType	D1IL
Origin	externalId	the origin element in "Plain" XML (incoming file name)
N/A	serviceProviderExternalId	L+G This is a constant. There is no such field in incoming structure.
RawData	rawData	Vendor-specific "raw" data When the "populateRawIMD" parameters in the EnvironmentSettings.xq file is set to true.

Mapping Additional Fields

Measurement and device event files received from the Landis+Gyr head-end system can include additional fields containing data to be imported into Oracle Utilities Smart Grid Gateway. These additional fields must be mapped to elements within the XML document processed by OSB and sent to Smart Grid Gateway.

This mapping can be performed through use of a custom XQuery document, specified in the EnvironmentSettings.xq file via the "modifyResultXMLInput" parameter.

The following sample XQuery documents illustrate how additional fields can be mapped into the XML format sent to Smart Grid Gateway.

Sample XQuery — Initial Measurements

The following XQuery is an example that shows a transformation that passes in a root element with 3 children (the “result” XML, the “plain” XML, the environment settings) that returns a modified “result” XML For testing purposes, it changes the original value in the `<enQty>` “result” element and replaces it with a value from the “plain” XML depending on an environment setting variable. The `<serviceProviderExternalId>` value was also replaced by a hard-coded value.

```
declare namespace lan = "http://xmlns.oracle.com/LandisGyrUsage";
declare namespace xf = "http://tempuri.org/D3/lgimd";
declare namespace soap = "http://schemas.xmlsoap.org/soap/envelope/";
declare function xf:modifyResultXML($modifyResultXMLInput as element(*)) as element(*){
<InitialLoadIMDList>
{
  for $InitLoadIMD in $ modifyResultXMLInput/InitialLoadIMDList/InitialLoadIMD
  return
    <InitialLoadIMD>
      <preVEE>
        <dvcIdN>{ data($InitLoadIMD/preVEE/dvcIdN) }</dvcIdN>
        <externalId>{ data($InitLoadIMD/preVEE/externalId)}</externalId>
        <uom>{ data($InitLoadIMD/preVEE/uom)}</uom>
        <mcIdN>{ data($InitLoadIMD/preVEE/mcIdN)}</mcIdN>
        <enDt>{ data($InitLoadIMD/preVEE/enDt)}</enDt>
        {
        if ($modifyResultXMLInput/EnvironmentSettings/test1="true")
        then <enQty>{ data($modifyResultXMLInput/lan:MeterReads/lan:MeterRead/lan:ExtraFields/
lan:ExtraField[lan:FieldName
= 'EF4']/lan:FieldValue) }</enQty>
         else <enQty>{ data($modifyResultXMLInput/lan:MeterReads/lan:MeterRead/lan:ExtraFields/
lan:ExtraField[lan:FieldName= 'EF2']/lan:FieldValue) }</enQty>
         }
       <imdType>{ data($InitLoadIMD/preVEE/imdType) }</imdType>
      </preVEE>
      <serviceProviderExternalId>NewSPId</serviceProviderExternalId>
    </InitialLoadIMD>}</InitialLoadIMDList>
};
declare variable $modifyResultXMLInput as element(*)external;
xf:modifyResultXML($modifyResultXMLInput)
```

Sample XQuery — Device Events

The following XQuery is an example that shows a transformation that passes in a root element with 3 children (the "result" XML, the "plain" XML, and the environment settings) and returns a modified "result" XML For testing purposes, it changes the original value in the `<externalCommunicationModuleIdentifier>` "result" element and replaces it with a value from the "plain" XML depending on an environment setting variable. The `<externalServiceLocationId>` value is also replaced by a hard-coded value.

```
declare namespace lan = "http://xmlns.oracle.com/LandisGyrEvent";
declare namespace xf = "http://tempuri.org/D3/event";
declare namespace soap = "http://schemas.xmlsoap.org/soap/envelope/";
declare function xf:modifyResultXML($modifyResultXMLInput as element(*)) as element(*){
  <DeviceEventSeeder>
```

```
    <externalSenderId>{data($modifyResultXMLInput/DeviceEventSeeder/externalSenderId) }</
externalSenderId>
    <deviceIdentifierNumber>{ data($modifyResultXMLInput/DeviceEventSeeder/
deviceIdentifierNumber) }</deviceIdentifierNumber>
    <externalEventName>{ data($modifyResultXMLInput/DeviceEventSeeder/externalEventName) }</
externalEventName>
    <eventDateTime>{ data($modifyResultXMLInput/DeviceEventSeeder/eventDateTime) }</
eventDateTime>
    <externalSourceIdentifier>{ data($modifyResultXMLInput/DeviceEventSeeder/
externalSourceIdentifier)
}</externalSourceIdentifier>
    <eventInformation>
      <externalEventCategory>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalEventCategory)
}</externalEventCategory>
      <externalEventSeverity>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalEventSeverity)
}</externalEventSeverity>
      <externalDeviceType>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalDeviceType)
}</externalDeviceType>
      <externalServiceLocationId>{1234 }</externalServiceLocationId>
      {
      if ($modifyResultXMLInput/EnvironmentSettings/testA="true")
      then <externalCommunicationModuleIdentifier>{ data($modifyResultXMLInput/lan:DeviceEvents/
lan:DeviceEvent/lan:DeviceType)}</externalCommunicationModuleIdentifier>
      else <externalCommunicationModuleIdentifier>{data($modifyResultXMLInput/lan:DeviceEvents/
lan:DeviceEvent/lan:CategoryId)}</externalCommunicationModuleIdentifier>
      }
     <externalStatusValue>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalStatusValue)
}</externalStatusValue>
      <externalStatusDateTime>{data($modifyResultXMLInput/DeviceEventSeeder/eventInformation/
externalStatusDateTime)}</externalStatusDateTime>
    </eventInformation>
  </DeviceEventSeeder>
};
declare variable $modifyResultXMLInput as element(*) external;
xf:modifyResultXML($modifyResultXMLInput)
```

MV90 for Itron

The Oracle Utilities Smart Grid Gateway MV-90 Adapter for Itron uses Oracle Service Bus (OSB) to facilitate communication between Oracle Utilities Smart Grid Gateway and MV-90. The following functionality is included:

Measurement Data Loading - Data parsing and transformation from MV-90 binary format into the Oracle Utilities Service and Measurement Data Foundation unified format.

Measurement Data Processing - Configurable mapping for MV-90 status codes to Oracle Utilities Service and Measurement Data Foundation standard values, along with configurable device event creation based on MV-90 status codes.

The following table describes the attributes of the adapter:

Attribute	Details
Currently Supported Version	SGG supports the .mv9 binary, mainframe data format.
Protocol	Binary file format.
Market(s)	Worldwide, but largely USA.
Architecture	AMR

MV90 Adapter Processing

This section provides details concerning the OSB processing and OUAF objects supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data Loading

The initial measurement data load and subsequent device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway service format. Payloads contain measurements in a head-end specific format OSB then places each service call into a JMS queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel then a service creates initial measurements with data in a common format with head-end-specific processing as needed.

Initial Measurements

The usage data exported from the AMI head-end system as a file in MV-90 format is loaded into Oracle Utilities as Initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. **SGG-D5-USAGE-BASE** contains components responsible for the actual processing of incoming data. It should not be modified during configuration. This project can be upgraded without affecting the customization and environment settings added to the SGG-D5-USAGE-CM project.
2. **SGG-D5-USAGE-CM** allows for customization and simplifies future upgrades.

The runtime configuration settings for the SGG-D5-USAGE-CM project are stored in the xquery file EnvironmentSettings.xq. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRawIMD	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterUsage	Determines if usage should be filtered.	true false
usePhysicalChannel	Optional value. Determines whether the physical channel ID is passed to the IMD seeder to create the measuring component identifier number. If this is set to false, the MV90 LOGCHAN field is used.	true false (default)
fieldForDvcIdN	Optional value. Specifies which field is used as the value for the device ID.	DC_RECID DC_CUSTID DC_METERID (default)

Element	Description	Valid Values
MV90ScalarChannelSuffix	Optional value. Holds a suffix value to be added to the measuring component identifier number when a scalar IMD is created for register reads. Default value is "_S"	
processMV90ScalarData	Optional value. Determines if register reads are processed.	true false (default)

Publishing Initial Measurement Data

The SGG adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Initial Measurement Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D5-USAGE-CM OSB project are used in publishing measurement data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Initial Measurement Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Initial measurement data is published in the "native" initial measurement data format (the format of the initial measurement seeder business object). This format includes normalized unit of measure and condition codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published through this feature is published prior to validation, estimation, and editing (VEE) processing. In addition, filtering *cannot* be applied to data published through this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288

Parameter	Description	Default Value
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Usage

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **MV90 UOM Code to Standard UOM Mapping** extendable lookup (D5-UOMtoStdUOMCodeMapLookup) are passed into the system for processing.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service for the project. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Prioritized Initial Measurement Processing

The SGG adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to “Real Time” (D1RT). In addition, initial measurements received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Initial Measurement Data Load with Register Reads

In the MV90 format, measurement data can include interval data as well as register reads that can be used to create scalar measurements. For information on configuring the adapter for loading scalar data based on register reads, see Override Processing Methods for Scalar Initial Measurement Data.

Base Package Business Objects

The MV-90 Adapter for Itron base package includes the following initial measurement business objects:

Business Object Name	Description
D5-InitialLoadIMDInterval	MV-90 Initial Load IMD

Business Object Name	Description
D5-InitialLoadIMDScalar	MV90 - Initial Load IMD - Scalar

Configuring an MV90 Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway MV-90 Adapter for Itron to receive usage from an MV-90 system.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system, including incoming usage.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-InitialLoadIMD	IMD Seeder
D1-PayloadErrorNotif	Payload Error Notification
D1-PayloadStatistics	Payload Statistics
D1-PayloadSummary	Payload Summary

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object
- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the MV-90 system must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System - MV-90:

External System: MV90

Description: MV90 - Mainframe File Format

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the MV-90 system must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Meter Data Management / Smart Grid Gateway documentation for more information about creating service providers.

Service Provider - MV-90:

- **Service Provider**: MV90
- **Description**: Head-End System for MV90
- **External Reference ID**: MV90
- **External System**: MV90 - Mainframe File Format
- **Out Name/ID in Their System**: MV90
- **AMI Device ID Type**: Serial Number
- **AMI Measuring Component ID Type**:

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to a given head-end system.

The following types of processing methods must be configured for the MV90 service provider. Refer to the Oracle Utilities Meter Data Management / Smart Grid Gateway documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head-end system.

Override Processing Methods for Scalar Initial Measurement Data

In the MV-90 format, meter read start and stop values can represent either a calculated read based on the interval data that is included in the MV90 data, or a register read that has been logged using a separate channel. When the values represent a register read the application can create a second file for the scalar initial measurements. The creation of a scalar initial measurement file is triggered by the Register Type flag, DC_REGTYPE, in the MV-90 data and the processMV90ScalarData variable in the EnvironmentSettings.xq file. The following register read values are valid for the DC_REGTYPE field:

Value	Description
V	Visual energy readings
D	Visual demand and energy readings
E	Encoder register readings (energy only).
B	Both encoder and visual energy readings.

Value	Description
C	Calculate stop meter readings from the encoder base reading

In order to configure your adapter to create scalar initial measurements based on register reads, first define new measuring component types and register measuring components. Then use the new measuring component types to specify override processing methods on the Initial Measurement Creation processing method for your service provider.
Measuring Component Types for Scalar Reads: To create initial measurement data for register reads, first create new measuring component types to represent the registers. These will be used to create measuring components for scalar reads.

Example Measuring Component Type

- **Measuring Component Type:** MV90_SCALAR_REG
- **Description:** MV90 Scalar Register
- **Measuring Component Business Object:** Register
- **Measurement Business Object:** Measurement
- **Service Type:** Electric Service
- **Allow Negative Consumption:** Allowed
- **Consumptive/Subtractive:** Subtractive
- **Read Method:** Automatic Read

Other attributes such as Value Identifiers, VEE Groups, and so on, should be defined based on requirements

Register Measuring Components for Scalar Reads: Use the new measuring component types to create measuring components for the scalar measurements created by the register read process. You must create a corresponding register measuring component for each interval channel for which you will receive scalar reads. The channel number for the registers must be based on the channel numbers of the corresponding interval channels, plus the suffix defined in the "MV90ScalarChannelSuffix" environment setting. The default value for this setting is "_S". These registers must also be added to the device configuration used by the interval channels (in other words, both measuring components - interval and scalar - must be on the same device). The following table shows examples of these values for the measuring components:

Device Type/Serial Number	MV-90 Interval Channel Number	MV-90 Scalar Register Channel Number
MV90 Electric/00001	1	1_S
MV90 Electric/00002	1	1_S
MV90 Electric/00003	1	1_S

Override Processing Methods for MV-90 Service Provider: Add an override processing method to the Initial Measurement Creation processing method for the MV90 service provider for each register measuring component type. The business object for each override processing method should be "D5-InitialLoadIMDScalar". The following tables shows examples of the override processing methods:

Measuring Component Type	Business Object
MV90 Scalar Register	D5-InitialLoadIMDScalar
Measuring Component Type 2	D5-InitialLoadIMDScalar
Measuring Component Type n	D5-InitialLoadIMDScalar

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.

UOM Translation

UOM translation processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map MV90 UOM codes to standard UOM codes when receiving usage from the MV-90 system.

Configuring MV90 Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway MV90 Adapter for Itron.

This section outlines some of the extendable lookups that must be configured for use with the MV-90 Adapter for Itron. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Interval Status Code to Event Mapping

The MV-90 adapter for does not accept device events from an MV-90 system, but can create device events based on specific interval status codes in an incoming usage reading. The MV-90 Interval Status Code to Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when creating device events based on interval status codes received with usage from the MV-90 system.

Each value defined for the Interval Status Code to Event Mapping extendable lookup should include the following:

- **MV90 Interval Status**: The interval status code used by the MV-90 system
- **Description**: A description of the MV-90 interval status code
- **Event Duration Mode**: The duration type for the event. Can be either “Individual” or “Continuous” (used for events with a duration, such as an outage).
- **Device Event Type**: The Device Event Type for the device event created for this interval status code. For status codes with an Event Duration Mode or “Continuous” this is the start event, or the first of the paired events created for this status code.
- **End Event Type**: The Device Event Type for the “end” device event created for this interval status code. For status codes with an Event Duration Mode or “Continuous” this is the start event, or the last of the paired events created for this status code.

Example: The MV-90 “Low Voltage” status code could be configured to create a “Low Voltage” device event as follows:

- **MV90 Interval Status**: <MV-90 interval status code for low voltage>
- **Description**: Low Voltage detected for meter
- **Event Duration Mode**: Individual
- **Device Event Type**: Low Voltage
- **End Event Type**: N/A

Example: The MV-90 “Power Outage” status code could be configured to create a “Last Gasp” device event, and a “Power Restoration” event as follows:

- **MV90 Interval Status**: <MV-90 interval status code for outage>

- **Description**: Outage
- **Event Duration Mode**: Continuous
- **Device Event Type**: Last Gasp
- **End Event Type**: Power Restoration

UOM Code to Standard UOM Code Mapping

Usage received from a utility's head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The UOM Code to Standard UOM Mapping extendable lookup is used for this purpose. Each value defined for the UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-end UOM**: The unit of measure code used by the head-end system.
- **Unit of Measure**: The unit of measure defined in the system.
- **Description**: A description of the unit of measure code.

Other Extendable Lookups

Business Object Name	Description
D5 - ChanelStatusCodeLookup	MV-90 Channel Status Code Lookup
D5-IntStsCdToCndMapPrLookup	Condition Mapping with Priority

Networked Energy Services

The Oracle Utilities Smart Grid Gateway Adapter for Networked Energy Services (NES) supports two-way communication with the NES server. Communications include measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, and on-demand read. The adapter uses Oracle Service Bus (OSB) and Oracle Business Process Execution Language Process Manager (BPEL PM) to facilitate communication between Oracle Utilities Smart Grid Gateway and NES. The following table describes the attributes of the adapter:

Attribute	Details
Currently Supported Version	Networked Energy Services 5.3
Protocol	Proprietary (NES specific)
Market(s)	Worldwide
Architecture	ANSI and IEC, PLC

Networked Energy Services Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, and Oracle Utilities Application Framework (OUAF) objects that are supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data and Device Event Loading

The initial measurement data load and device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway (SGG) service format. Payloads contain measurements and meter events in some head-end specific format. OSB then places each service call into a Java Message Service (JMS) queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel. A service

then creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format.

Initial Measurements

Usage data is exported from the head-end system as a file in NES format and is loaded into Oracle Utilities as initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. **SGG-D4-USAGE-BASE** contains components responsible for the actual processing of incoming data. This should not be modified during configuration.
2. **SGG-D4-USAGE-CM** allows you to segment your customizations so that future upgrades of base functionality implemented in SGG-D4-USAGE-BASE do not affecting the customization and environment settings.

The runtime configuration settings for the SGG-D4-USAGE-CM project are stored in the xquery file EnvironmentSettings.xq. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	
destinationRootElementScalar	Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterUsage	Determines if usage should be filtered.	true false

Publishing Initial Measurement Data

The SGG adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Initial Measurement Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D4-USAGE-CM OSB project are used in publishing measurement data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.

- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Initial Measurement Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Initial measurement data is published in the "native" initial measurement data format (the format of the initial measurement seeder business object). This format includes normalized unit of measure and condition codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published through this feature is published prior to validation, estimation, and editing (VEE) processing. In addition, filtering *cannot* be applied to data published through this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Initial Measurement Data

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to "true" as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **NES UOMCode Mapping Extendable Lookup** extendable lookup (D4-HeadendUOMLookup) are passed into the system for processing.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service for the project. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Prioritized Initial Measurement Processing

The SGG adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to “Real Time” (D1RT). In addition, initial measurements received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Device Events

The device event data exported from the head-end system as a file in NES format is loaded into Oracle Utilities as a Device Event. One of your configuration tasks is to customize the device events processing.

The required functionality is delivered in the base product as two OSB projects:

1. **SGG-D4-EVENT-BAS**E contains components responsible for the actual processing of incoming data. This should not be modified during configuration.
2. **SGG-D4-EVENT-CM** allows you to segment your customizations so that future upgrades of base functionality implemented in SGG-D4-EVENT-BASE do not affecting the customization and environment settings.

The runtime configuration settings for the SGG-D4-EVENT-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the device event data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElement	Holds the name of inbound web service for the device event seeder.	
serviceProviderExternalId	The External ID of the NES service provider.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false

Publishing Events

The NES adapter can be configured to publish device events for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Device Event Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D4-EVENT-CM OSB project are used in publishing device events data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Device Event Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Device event data is published in the "native" device event data format (the format of the device event seeder business object). This format includes normalized device event codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: In addition, filtering can NOT be applied to device events published via this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging

Parameter	Description	Default Value
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Events

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **NES Device Event Mapping** extendable lookup (D4-DeviceEventMappingLookup) are passed into the system for processing.

Prioritized Device Event Processing

The SGG adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Configuring Payload Processing Notifications

By default, Payload Statistics, Payload Error Notification, and Payload Summary notification activities are sent to the Notification Queue within Oracle Service Bus. Smart Grid Gateway adapters can be configured to restrict sending these activities by defining optional JCA transport dynamic properties in the appropriate OSB project. The following optional JCA transport dynamic properties can used to control if these notification activities are sent:

- **SGG_SEND_STATISTICS**: Controls sending Payload Statistic activities (true: send, false: do not send)
- **SGG_SEND_ERROR**: Controls sending Payload Error Notification activities (true: send, false: do not send)
- **SGG_SEND_SUMMARY**: Controls sending Payload Summary activities (true: send, false: do not send)

When these properties are set to false, the corresponding payload notification will not be sent to the JMS Queue. Note these properties are not populated by default, and default to true if omitted.

To create these properties:

1. Navigate to the SGG-D4-EVENT-CM project in the Oracle Service Bus Console.
2. Expand the **Proxy Services** folder.
3. Select the **InboundProxyService** proxy service (not the Pipeline or WSDL).
4. Click **Create** to create a new session.
5. Select **Transport Details**.
6. Click the plus sign in the **Dyanmic EndPoint Properties** section. An empty row will appear.
7. Enter the property you wish to define in the **Property** column.
8. Enter "false" in the **value** column.
9. Repeat steps 6-8 for each property you wish to define.
10. Click **Activate** to save and activate your changes.

NOTE: These properties are only used if the SGG_EXT_ID_PATTERN is not present or the regular expression captures nothing. The SGG_EXT_ID_PATTERN will override these properties when the regular expression captures group data.

Base Package Business Objects

The NES adapter base package includes the following device, initial measurement, and device event business objects:

Business Object Name	Description
D4-InitialLoadIMDInterval	NES Initial Load IMD Interval
D4-InitialLoadIMDScalar	NES Initial Load MD Scalar
D4-SmartMeter	Smart Meter - NES
D4-EventExtractScheduler	NES Event Extract Scheduler
D4-PayloadExtractScheduler	Payload Extract Scheduler
D4-UsageExtractScheduler	NES Usage Extract Scheduler

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, an on-demand read, or another type of request that the head-end system supports. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related web service from the head-end system. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

Communication Flows

The table below lists the communications created for each NES command:

Command	Outbound Communication	Inbound Communication	Completion Event
Remote Connect	Retrieve Meter Information Read Load Status Read Billing Data On-Demand Connect Load	Read Load Status Notification Read Billing Data On-Demand Notification Connect Load Notification	Create IMD Connect Device
Remote Disconnect	Disconnect Load Read Billing Data On-Demand	Disconnect Load Notification Read Billing Data On-Demand Notification	Create IMD Disconnect Device
Device Commissioning	Add Meter Request Set ATM Configuration	Set ATM Configuration Notification	Device Commissioning
Device Decommissioning	Decommission Meter Request	Decommission Meter Notification	Device Decommissioning
On-Demand Read (Scalar)	Read Billing Data On-Demand	Read Billing Data On-Demand Notification	Create IMD
On-Demand Read (Interval)	Read Load Profile On Demand	Read Load Profile On-Demand Notification	Create IMD

Commissioning and Decommissioning Communications

Commissioning a previously decommissioned NES device requires a different process than commissioning a new device.

When you commission a device for the first time, a commissioning command is sent to BPEL to create the meter in NES using the DeviceManager.CreateMeter command. If Automatic Topology Management is selected then the DeviceManager.SetATMConfiguration command is also sent.

When you decommission a device, BPEL sends the DeviceManager.Move command to NES for the device. In NES you need to confirm that the device has been moved to a non-DC tree. You then need to manually delete the device from the non-DC tree. To permanently decommission the device you must delete it by using the DeviceManager.Delete command from the NES interface.

To recommission a device that has been previously decommissioned, you need to log into NES and manually move the meter from the non-DC tree to the DC tree that it belonged to previously. This will not be possible if the device has been deleted from within NES.

Device Communication Base Package Business Objects

The NES base package includes the following communication business objects:

Meter Commissioning

Business Object	Description
D4-AddMeterRequest	Add Meter Request
D4-GenericAMIDeviceIdentifier	Generic AMI Device Identifier
D4-RetrieveMeterIdentifier	Retrieve Meter Identifier
D4-RetrieveMeterInfo	Retrieve Meter Information
D4-SetATMConfigNotification	Set ATM Configuration Notification
D4-SetATMConfiguration	Set ATM Configuration

Meter Decommissioning

Business Object	Description
D4-DecommMeterNotification	Decommission Meter Notification
D4-DecommMeterRequest	Decommission Meter Request

On-Demand Read

Business Object	Description
D4-ReadBillingData	Read Billing Data On-Demand
D4-ReadBillingDataNtf	Read Billing Data On-Demand Notification
D4-ReadLoadProfile	Read Load Profile On Demand
D4-ReadLoadProfileNtf	Read Load Profile On-Demand Notification
D4-ReadLoadStatus	Read Load Status
D4-ReadLoadStatusNtf	Read Load Status Notification

Remote Connect

Business Object	Description
D4-ConnectLoad	Connect Load
D4-ConnectLoadNtf	Connect Load Notification

Remove Disconnect

Business Object	Description
D4-DisconnectLoad	Disconnect Load
D4-DisconnectLoadNtf	Disconnect Load Notification

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch or Real-time)
- The corresponding message senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgment and response messages are sent and received validating that commands have been transmitted.

Outbound Message Type	Description
D4-ADDMETREQ	Add Meter Request
D4-CONLOAD	Connect Load
D4-DECMETREQ	Decommission Meter Request
D4-DISCONLD	Disconnect Load
D4-HNLBATEVT	NES - Handle Batch Event
D4-HNLBATUSG	NES - Handle Batch Usage
D4-RDLOADST	Read Load Status
D4-READLP	Read Load Profile
D4-READOUT	Read Billing Data
D4-RETMETID	Retrieve Meter Identifier
D4-RETRIEVE	Retrieve Meter Information
D4-SETATMCFG	Set ATM Configuration

Inbound / Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The Oracle Utilities Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway Adapter for NES uses the following inbound web services to import usage and device events:

Inbound Web Service	Description
D1-DeviceEventSeeder	Used for upload of device events. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-InitialLoadIMD	Used for initial measurement upload. The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.

The Oracle Utilities Smart Grid Gateway Adapter for NES includes the following inbound web services for receiving communications from the NES:

Inbound Web Service Name	Description
D4-ConnectLoadNotification	Connect Load Notification
D4-DecommMeterNotificationXAI	Meter Decommission Notification
D4-DisconnectLoadNotification	Disconnect Load Notification
D4-ReadBDNotificationXAI	Read Billing Data On-Demand Notification
D4-ReadingATMConfigNotification	Get Response Initiate Read By Meter Number
D4-ReadLoadStatusNotification	Read Load Status Notification
D4-ReadLPNotificationXAI	Read Load Profile On-Demand Notification

Message Senders

Message senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of message senders you need to create is based on the types of messages the system is designed to accept.

The Oracle Utilities Smart Grid Gateway Adapter for NES includes the following message senders:

Message Sender	Description
D4-ADDMETREQ	Add Meter Request
D4-CONLOAD	Connect Load
D4-DECOMMIS	Device Decommissioning
D4-DISCONLD	Disconnect Load
D4-HNLBATEVT	NES Handle Batch Event
D4-HNLBATUSG	NES Handle Batch Usage
D4-RDLOADST	Read Load status
D4-READBILLD	NES On-Demand Read (Scalar)(2)
D4-READLP	Read Load Profile
D4-RETMETID	Retrieve Meter Identifier
D4-RETRIEVE	Retrieve Meter Information
D4-ReadBill	Read Billing Data
D4-SETATMCFG	Set ATM Configuration

BPEL Processes and Web Services

These processes are responsible for performing the conversion from Oracle Utilities format to NES format, invoking process callouts and invoking the remote endpoint to trigger the device events.

All web services receive XML from an incoming service then transform the call to a format which is recognized by the NES system.

CreateMeter: Receives the incoming XML from the message sender service, transforms it to the NES API parameters node, passes it to the DeviceManager.Create NES API web service method then collects XML from CreateMeter to be returned to Oracle Utilities Smart Grid Gateway.

Messages

CreateMeter RequestMessage: Contains the parameters which are formatted from Oracle Utilities Smart Grid Gateway to match the incoming XML for the NES API DeviceManager.Create method.

CreateMeter ResponseMessage: Contains the Device ID that is used when the DeviceManager.Create method validates the security key and the XML parameters successfully.

Related Processes

Meter Commissioning

ConnectLoadRemoteConnect
Receives the incoming XML from the D4-CONLOAD (Connect Load) message sender in Oracle Utilities Smart Grid Gateway and transforms to NES format.

GrabBatchEvents
Retrieves the un-processed events from the event repository; transforms each batch to replace GUID identifiers with a readable form, and finally prepares a new file for OSB before deleting the batch of events. The scheduled call to this service can contain the preferred dates to query for or can accept the default which is the previous day usage.

Messages

GrabBatchEventsSoapIn

GrabBatchEventsSoapOut

GrabBatchUsage
Retrieves the un-processed usage from the usage repository; transforms each batch to replace GUID identifiers with a readable form, and finally prepares a new file for OSB before deleting the batch of usage. The scheduled call to this service can contain the preferred dates to query for or can accept the default which is the previous day usage.

Messages

GrabBatchUsageSoapIn

GrabBatchUsageSoapOut

Related Processes

Connection

Event Management

Usage Processing

MeterDecommissioning
Receives the incoming XML from the message senderOutbound service and transforms to NES API parameters node to be passed to the DeviceManager.Move NES API web service method. The process then collects the return XML to be returned to Oracle Utilities Smart Grid Gateway.

Messages

MeterDecommissioningRequestMessage: This message is the input for the Move method. The method contains the parameters conveniently formatted from Oracle Utilities Smart Grid Gateway to match the incoming XML for the NES API DeviceManager.Move method.

MeterDecommissioningResponseMessage: This message contains status stating that the DeviceManager.Create method validated the security key and the XML parameters are accepted successfully. If the status is SUCCEEDED, it also contains Tracking ID for the call, Device ID and StatusType ID.

Related Processes

Meter Decommissioning

PerformCommand
This web service facilitates most of the communication between Oracle Utilities Smart Grid Gateway and the NES system. If your implementation requires functionality that was not delivered as part of the base package this web service will play an integral part in managing those customizations.

PerformCommand executes a command which generates the events that advance the Oracle Utilities Smart Grid Gateway lifecycle then returns the output string to the caller. The output might be a tracking ID or information about a device.

Messages

PerformCommandSoapIn: The sXmlParameters input is prepared from within Oracle Utilities Smart Grid Gateway to execute primarily a group of five commands, these are Read Load Status, On-Demand Read, Connect Load, Disconnect Load and Set ATM Configuration.

PerformCommandSoapOut:

The return from PerformCommand is the xml from the DeviceManager.PerformCommand for Read Load Status, On-Demand Read, Connect Load, Disconnect Load and Set ATM Configuration.

ReceivePanoramixEvents
This web service is actually never called by the implementation, but rather it is called by NES and is needed when configuring the NES server.

ReceivePanoramixEvents obtains the APIKey from the NES authentication service, then, with the event definition ID from the parameter list for this service, switches to the branch for the event definition to begin any specific processing. For example, an on-demand read completion event means that the BPEL PM process needs to retrieve the usage using the RESULTID found in the event xml. All events and usage to be passed on to Oracle Utilities Smart Grid Gateway must

first be transformed to replace the GUID with a readable form items. This involves using the BPEL PM DVM lookup functionality.

Messages

ReceivePanoramixEventsSoapIn

ReceivePanoramixEventsSoapOut

Related Processes

Event Management

PrePerformCommandCallout / PostPerformCommandCallout
These web services are hooks designed to allow customization of messages just prior and just after sending them to the NES server. Your implementation may wish to modify the data for special purposes. This is the one of the only places where modification of BPEL PM code is allowed.

ReceivedPanoramixEventsCallout
This web service is a hook designed to allow customization of messages just prior and just after sending them to the NES server. Your implementation may wish to modify the data for special purposes. This is one of the only places where modification of BPEL PM code is allowed.

ReadBillingDataOnDemand
This web service method, initiate a call to NES system with formatted inbound xml parameter. First it transforms the Parameters received from the calling system in to NES system recognizable format. Then it invokes a call to the NES system's Device Manager Service. When the call is invoked the system receives the return XML from DeviceManager stating the status of the command execution and, if succeeded, the tracking ID of this call.

Messages

ReadBillingDataOnDemandRequestMessage: Contains the input parameter for DeviceManager Perform Command to read billing data OnDemand. It formats parameters from the calling system to match incoming XML for the NES DeviceManager.PerfomCommand method.

ReadBillingDataOnDemandResponseMessage: Transforms the response xml from NES system format to Oracle Utilities Smart Grid Gateway format.

Related Processes

On Demand Reads

ReadLoadProfileOnDemand
This web service method, initiate a call to NES system's Device Manager Service with formatted inbound xml parameter. First it transforms the Parameters received from the calling system in to NES system recognizable format. Then it invokes a call to the NES system thru Device Manager Service. After invoking the call, the system receives the response message from DeviceManager stating the status of the command execution and, if succeeded, the tracking ID of this call.

Messages

ReadLoadProfileOnDemandRequestMessage: This message is for interval on demand read request to the NES system. Transforms the incoming XML from Oracle Utilities Smart Grid Gateway to NES recognizable format.

ReadLoadProfileOnDemandResponseMessage: Transforms the response xml for ReadLoadProfileOnDemandRequest from NES system format to Oracle Utilities Smart Grid Gateway format.

Related Processes

On Demand Reads

ReadLoadStatusRemoteConnect
Receives the incoming XML from the D4-RDLOADST (Read Load Status) message sender in Oracle Utilities Smart Grid Gateway and transforms to NES format.

RemoteDisconnect

Receives the incoming XML from the D4-DISCONLD (Disconnect Load) message sender in Oracle Utilities Smart Grid Gateway and transforms to NES format.

RetrieveMeterInfo
Initiates a call to the Request Management Service with formatted inbound xml parameters. First it transforms the Parameters received from the calling system in to NES system recognizable format. Then it invokes a call to the NES system thru Request Management Service and retrieves a response with Meter Information.

Messages

RetrieveMeterInformationRequestMessage: Contains the input parameter for DeviceManager Perform Command to retrieve meter information. It also formats parameters from the calling system to match incoming XML for the NES DeviceManager.PerfomCommand method.

RetrieveMeterInformationResponseMessage: Transforms the response xml from NES system format to Oracle Utilities Smart Grid Gateway format.

Related Processes

Remote Disconnect

Remote Connect

RetrieveMeterIdentifier
Receives a call from Oracle Utilities Smart Grid Gateway for RetrieveMeterIdentifier process and invokes the RetrieveByParameter method in the NES system's DeviceManager service.

Converts incoming Oracle Utilities Smart Grid Gateway formatted input to sXmlParameters in NES format then invokes RetrieveMeterIdentifier.

Messages

RetrieveMeterIdentifierRequestMessage: Transforms the xml received from the calling system to match incoming XML for the NES API DeviceManager.RetrieveByParameter method. This method requires the Type of ID and the associated ID as input.

RetrieveMeterIdentifierResponseMessage: When the XML is synchronously returned from the head-end system for the RetrieveByParameter call, the DEVICEID node is only included in the APIPAYLOAD if the status of the call is Succeeded.

Related Processes

Meter Commissioning

SetATMConfiguration
This web service method is specific to set Automated Topology Management (ATM) configuration command which assigns the Meter to the Data Concentrator.

Converts incoming Oracle Utilities Smart Grid Gateway formatted input to sXmlParameters in NES format then invokes the DeviceManager PerformCommand. A connection should establish to begin the NES command processing.

Messages

SetATMConfigurationRequestMessage: This message is the input for the SetATMConfiguration method. The method contains the parameters conveniently formatted from Oracle Utilities Smart Grid Gateway to match the incoming XML for the NES API DeviceManager.PerformCommand method where the COMMANID is Constants.DeviceCommands.SET_ATM_CONFIGURATION

SetATMConfigurationResponseMessage: when the XML is synchronously returned from the head-end system for the PerformCommand call, the COMMAND node is only included in the APIPAYLOAD if the status of the call is Succeeded.

Related Processes

Meter Commissioning

NES Web Services

The following table describes NES web services and operations used for the Oracle Utilities Smart Grid Gateway command messaging:

Smart Grid Gateway Commands	AMI Adapter Business Objects	NES Web Services	NES Operations
Device Commissioning	D4-AddMeterRequest D4-RetrieveMeterIdentifier D4-SetATMConfiguration	DeviceManager	Create RetrieveByParameter PerformCommand: SET_ATM_CONFIGURATION
Device Decommissioning	D4-DecommMeterRequest	DeviceManager	PerformCommand: MOVE_DEVICE_ADD
Remote Connect	D4-ConnectLoad D4-ReadLoadStatus D4-RetrieveMeterInfo	DeviceManager	Retrieve PerformCommand: CONNECT_LOAD READ_LOAD_STATUS
Remote Disconnect	D4-DisconnectLoad	DeviceManager	PerformCommand: DISCONNECT_LOAD
On-Demand Read	D4-ReadBillingData	DeviceManager	PerformCommand Command ID: READ_BILLING_DATA_ON_DEMAND (Scalar) READ_LOAD_PROFILE_ON_DEMAND (Interval)

Configuring a Networked Energy Services Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway NES Adapter to communicate with the NES software.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the NES in response to a command request.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Device Event Seeder
D1-InitialLoadIMD	IMD Seeder
D1-PayloadErrorNotif	Payload Error Notification

Inbound Web Service Name	Description
D1-PayloadStatistics	Payload Statistics
D1-PayloadSummary	Payload Summary
D4-ConnectLoadNotification	Connect Load Notification
D4-DecommMeterNotificationXAI	Meter Decommission Notification
D4-DisconnectLoadNotification	Disconnect Load Notification
D4-ReadBDNotificationXAI	Read Billing Data On-Demand Notification
D4-ReadingATMConfigNotification	Get Response Initiate Meter Read By Meter Number
D4-ReadLoadStatusNotification	Read Load Status Notification
D4-ReadLPNotificationXAI	Read Load Profile On-Demand Notification

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object
- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

Message Senders

Message senders define the details of how messages are sent to an external system, such as messages containing device command requests.

The following message senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders.

Message Sender	Description
D4-ADDMETREQ	Add Meter Request
D4-CONLOAD	Connect Load
D4-DECOMMIS	Device Decommissioning
D4-DISCONLD	Disconnect Load
D4-HNLBATEVT	NES Handle Batch Event
D4-HNLBATUSG	NES Handle Batch Usage
D4-RDLOADST	Read Load status
D4-READBILLD	NES On-Demand Read (Scalar)(2)
D4-READLP	Read Load Profile
D4-RETMETID	Retrieve Meter Identifier
D4-RETRIEVE	Retrieve Meter Information
D4-ReadBill	Read Billing Data
D4-SETATMCFG	Set ATM Configuration

Note: The following apply to all of the above message senders:

Main Tab:

- **Invocation Type**: Real-time

- **Message Class**: RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction: http://xmlns.oracle.com/ouaf/echelon/**<OPERATION>**
- **HTTP Login User**: **<USER_ID>**
- **HTTP Login Password**: **<PASSWORD>**
- **HTTP Method**: POST
- **HTTP URL 1**: http://**<EM_SERVER>**:**<EM_SERVER_PORT>**/soa-infra/services/D4/**<SERVICE>**/**<SERVICE>**

  where:

  - **<OPERATION>**: the operation performed by the message sender (see Operation column in the table above)
  - **<USER_ID>**: the user ID used to log into WebLogic Enterprise Manager
  - **<PASSWORD>**: the password used to log into WebLogic Enterprise Manager
  - **<EM_SERVER>**: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
  - **<EM_SERVER_PORT>**: the port where the WebLogic Enterprise Manager is installed
  - **<SERVICE>**: the service invoked by the message sender (see Service column in the table above)

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
D4-ADDMETREQ	Add Meter Request
D4-CONLOAD	Connect Load
D4-DECMETREQ	Decommission Meter Request
D4-DISCONLD	Disconnect Load
D4-HNLBATEVT	NES - Handle Batch Event
D4-HNLBATUSG	NES - Handle Batch Usage
D4-RDLOADST	Read Load Status
D4-READLP	Read Load Profile
D4-READOUT	Read Billing Data
D4-RETMETID	Retrieve Meter Identifier
D4-RETRIEVE	Retrieve Meter Information
D4-SETATMCFG	Set ATM Configuration

Note: The following apply to all of the above outbound message types:

- **Business Object**: D1-OutboundMessage (Outbound Message)
- **Priority**: Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the NES must be present in your system. If this is not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System - NES:

- **External System**: NES
- **Description**: NES
- **Outbound Message Types**::

Outbound Message Type	Message Sender
D4-ADDMETREQ	D4-ADDMETREQ
D4-CONLOAD	D4-CONLOAD
D4-DECMETREQ	D4-DECOMMIS
D4-DISCONLD	D4-DISCONLD
D4-HNLBATEVT	D4-HNLBATEVT
D4-HNLBATUSG	D4-HNLBATUSG
D4-RDLOADST	D4-RDLOADST
D4-READLP	D4-READBILLD
D4-READOUT	D4-READLP
D4-RETMETID	D4-RETMETID
D4-RETRIEVE	D4-RETRIEVE
D4-SETATMCFG	D4-SETATMCFG

Note: The following apply to all of the above outbound message types:

- **Processing Method**: Real-time
- **Message XSL**: D4-Request xsl
- **Response XSL**: D4-Response.xsl

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the NES must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about creating service providers.

Service Provider - NES

- **Service Provider**: NES
- **Description**: NES
- **External Reference ID**: NES

- **External System**: D4-NES
- **Out Name/ID in Their System**:
- **AMI Device ID Type**: Internal Meter Number
- **AMI Measuring Component ID Type**: Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to a given head-end system.

The following types of processing methods must be configured for the NES service provider. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head-end system.

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.

UOM Translation

UOM mapping processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the NES service provider, based on the requirements of each implementation using the “How to Create OB COMM/Send OB Message” processing method business object (D1–HowToCreateActivityOBComm).

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Registration	D4-SetATMConfiguration	Set ATM Configuration
Device Decommission	Device Removal	D4-DecommMeterRequest	Decommission Meter Request
On-Demand Read (Scalar)	On-Demand Read (Scalar)	D4-ReadBillingData	Read Billing Data
On-Demand Read (Interval)	On-Demand Read (Interval)	D4-ReadLoadProfile	Read Load Profile
Device Status Check	Device Status Check	D4-ReadLoadStatus	Read Load Status
Remote Connect	Remote Connect	D4-ConnectLoad	Connect Load

Command	Processing Role	Default Business Object	Default Outbound Message Type
Remote Disconnect	Remote Disconnect	D4-DisconnectLoad	Disconnect Load

Other Processing Methods

This section outlines details for other processing methods used by the NES adapter.

Processing Role	Default Business Object	Default Outbound Message Type
NES Add Meter Request	D4-AddMeterRequest	Commission
NES Retrieve Meter Identifier	D4-RetrieveMeterIdentifier	Commission Retrieve Meter Identifier
NES Retrieve Meter Information	D4-RetrieveMeterInfo	Retrieve Meter Info

Configuring Endpoint URIs

Part of the configuration process is configuring your BPEL composites to work with your head-end system by defining the appropriate Endpoint URIs for each of the commands. The default approach to defining Endpoints URIs is redeployment or reinstallation or the BPEL composites. For example, by default, changing an adapter from using the test harness to a production environment using the actual head-end system requires editing the appropriate installation menu options and redeploying the BPEL composites. The Endpoints URIs defined during installation and deployment for each adapter are listed in the **Smart Grid Gateway Installation and Configuration Worksheets** section of the *Oracle Utilities Smart Grid Gateway Installation Guide*.

You can also use an “Endpoint Override” Domain Value Map (DVM) to override Endpoint URIs defined during deployment and installation. This DVM allows defining specific keys that provide an alternate URL that will override the original installed value. Endpoint Override DVMs can be edited using the Oracle SOA Composer.

The NES endpoint override DVM (D4–EndpointOverrides.dvm) uses a specific set of keys, each used for one or more commands. The table below lists the DVM keys available for the NES adapter and the command used with each:

DVM Key	Commands
DeviceManager	Remote Connect Remote Disconnect Device Commissioning Device Decommissioning On-Demand Read (Scalar) On-Demand Read (Interval)
EventManager	On-Demand Read (Scalar) On-Demand Read (Interval)
GatewayManager	Remote Connect Remote Disconnect Device Commissioning Device Decommissioning On-Demand Read (Scalar) On-Demand Read (Interval)
SettingManager	Remote Connect Remote Disconnect Device Commissioning Device Decommissioning On-Demand Read (Scalar)

DVM Key	Commands
	On-Demand Read (Interval)
UserManager	Remote Connect
	Remote Disconnect
	Device Commissioning
	Device Decommissioning
	On-Demand Read (Scalar)
	On-Demand Read (Interval)
ProcessCallout	User Exit Functions

To define an override Endpoint URI for the NES adapter, use the following procedure:

1. Open the SOA Composer for your BPEL configuration.

   The URL for the SOA Composer is http://server:port/soa/composer.

2. Select the D4–EndpointOverrides.dvm in the left panel.

   - In Fusion Middleware v12.2.1, this is located under **Shared** in the **Deployment View**, or under **Domain Value Maps** in the **Types View**.
   - In Fusion Middleware v12.2.2, this is located under **Metadata** in the **Deployment View**.

   The DVM will open in the right panel. The **Description** field lists the available keys for the DVM (only the first key is shown, but you can scroll through the contents to view the list. The panel also displays a list of previously defined keys.

3. Click the **Create Session** button (above the left panel) to begin an editing session.

4. Click the **Add Domain Values** icon ("+") to add a new key.

   The **Add Domain Values** dialog opens.

5. Enter the appropriate values in the **Add Domain Values** dialog as follows:

   - **key**: The DVM key for the Endpoint URI you wish to define (see the table above).
   - **EndpointURI**: The override Endpoint URI.

6. Click **OK**.

   The new DVM value will appear in the list of keys.

7. Click the **Save** icon to save the DVM values.

8. Click the **Publish** button to activate all the changes in the editing session.

   Enter an optional note for the session in the **Publish Session** dialog.

   Changes take effect immediately upon publishing the session.

   Click **Discard** to discard your changes.

   Click **Exit** to exit your current session. Note that your session will still be open if you exit. Use **Discard** to end your session without making changes.

Configuring Networked Energy Services Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Networked Energy Services.

This section outlines some of the extendable lookups that must be configured for use with the NES adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system. Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name**: The event name used by the head-end system
- **Description**: A description of the device event
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Standard Event Name**: The standard event name for device events of this type, from the "Standard Event Name" extendable lookup.

UOM Code to Standard UOM Mapping

Usage received from a utility's head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The UOM Code to Standard UOM Mapping extendable lookup is used for this purpose. Each value defined for the UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-end UOM**: The unit of measure code used by the head-end system.
- **Unit of Measure**: The unit of measure defined in the system.
- **Description**: A description of the unit of measure code.

Interval Status Code to Condition Mapping

Interval usage received from the NES can include NES interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The NES Interval Status Code to Condition Mapping extendable lookup is used to determine how to map NES interval status codes to standard status codes when receiving usage from the NES.

Each value defined for the NES Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status**: The NES interval status code
- **Condition**: The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.
- **Description**: A description of the interval status code.
- **Priority**: The priority of the interval status code, represented as a numeric value where larger numbers correlate to higher priorities.

Other Extendable Lookups

Business Object	Description
D4-EntityTypeLookup	Entity Type
D4-ExtServiceReturnCodeLookup	External Service Return Code
D4-GatewayTypeLookup	Gateway Type
D4-IDTypeLookup	ID Type
D4-InformationReturnTypeLookup	Information Return Type
D4-LoadVoltageStatusTypeLookup	Load Voltage Status Type
D4-TaskPriorityLookup	Task Priority

Business Object	Description
D4-TimeZoneManagerLookup	Time Zone Manager

Configuring NES Usage and Event Extract Processing

Usage and events received from the NES server must be requested from Oracle Utilities Smart Grid Gateway. Extract requests for usage and events are triggered by a batch process, and sent to the NES via middleware. The NES compiles a batch of usage or events and sends it to BPEL services to create a file to be processed by Oracle Service Bus (OSB) services, it is then loaded as SGG events and/or usage.

This section describes the processing performed by the extract request process, and the configuration steps involved.

Extract Request Processing

This section provides a detailed description of the extract request process.

- The “Usage / Event Extract Scheduler Monitor” batch process (D1-EXTSC) checks for “NES Usage Extract Scheduler” or “NES Event Extract Scheduler” activities that are in the “Active” state, and transitions these to the “Send Request” state. It also updates the “Latest Request Start Date/Time” and “Latest Request End Date/Time” fields on the activity based on the date and time of the request.
- When the activity enters the “Send Request” state, an Enter algorithm (either “Usage Extract Scheduler Send Request” (D4-USGSNDREQ) or “Event Extract Scheduler Send Request” (D4-EVTSNDREQ)) sends a request via an message sender to the middleware (via the GrabBatchUsage or GrabBatchEvent BPEL services), which in turn send the request to the NES server.
- After the request is sent, the “NES Usage Extract Scheduler” or “NES Event Extract Scheduler” activity to set back to the “Active” state.
- The NES server calls the BPEL services “GrabBatchEvents” and “GrabBatchUsage.” BPEL then routes the incoming events or usage to a file system, which is used by OSB components to process the usage or events.

Extract Request Configuration Steps

This section outlines the objects that must be configured to support batch extracts of usage and events from an NES head-end system. To configure the NES Adapter to support sending extract requests for usage and events, you must do the following:

- Create message senders (one for usage, one for events) to send extract requests.
- Create Outbound Message Types (one for usage, one for events) for extract requests.
- Associate the message senders and Outbound Message Types to the External System used with the head-end system service provider
- Create “NES Usage Extract Scheduler Type” and “NES Event Extract Scheduler Type” activity types.
- Create “NES Usage Extract Scheduler” and “NES Event Extract Scheduler” activities.

The following sections provide details for configuring these object. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders, Outbound Message Types, and External Systems.

Note: Configuration of the OSB and BPEL services used by this process is done when installing and configuring the OSB and BPEL components of the NES Adapter.

Message Sender - NES Handle Batch Event

Create an message sender for event extracts as follows:

Main Tab:

- **Message Sender**: Enter a code for the batch event message sender (D4-HNLBATEVT)
- **Description**: NES Handle Batch Event
- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes messages via HTTP real-time)
- **Active**: True (checked)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction:http://xmlns.oracle.com/ouaf/NES/GrabBatchEvents
- **HTTP Login User**: <**USER_ID**>
- **HTTP Login Password**: <**PASSWORD**>
- **HTTP Method**: POST
- **HTTP URL 1**: http://<**EM_SERVER**>:<**EM_SERVER_PORT**>/soa-infra/services/NES/HandleBatchEvent/BatchEventHandler

  where:

  - <**USER_ID**>: the user ID used to log into WebLogic Enterprise Manager
  - <**PASSWORD**>: the password used to log into WebLogic Enterprise Manager
  - <**EM_SERVER**>: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
  - <**EM_SERVER_PORT**>: the port where the WebLogic Enterprise Manager is installed

Message Sender - NES Handle Batch Usage

Create an message sender for usage extracts as follows:

Main Tab:

- **Message Sender**: Enter a code for the batch usage message sender (D4-HNLBATUSG)
- **Description**: NES Handle Batch Usage
- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes messages via HTTP real-time)
- **Active**: True (checked)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction:http://xmlns.oracle.com/ouaf/NES/GrabBatchUsage
- **HTTP Login User**: <**USER_ID**>
- **HTTP Login Password**: <**PASSWORD**>
- **HTTP Method**: POST
- **HTTP URL 1**: http://<**EM_SERVER**>:<**EM_SERVER_PORT**>/soa-infra/services/NES/HandleBatchUsage/BatchUsageHandler

  where:

  - <**USER_ID**>: the user ID used to log into WebLogic Enterprise Manager
  - <**PASSWORD**>: the password used to log into WebLogic Enterprise Manager

- **<EM_SERVER>**: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
- **<EM_SERVER_PORT>**: the port where the WebLogic Enterprise Manager is installed

Outbound Message Type - NES Handle Batch Event

Create an Outbound Message Type for event extract requests as follows:

- **Outbound Message Type**: Enter a code for the batch event Outbound Message Type (D4-HNLBATEVT)
- **Description**: NES Handle Batch Event
- **Business Object**: D1-OutboundMessage
- **Priority**: - Priority 20 (or lower)

Outbound Message Type - NES Handle Batch Usage

Create an Outbound Message Type for usage extract requests as follows:

- **Outbound Message Type**: Select a code for batch usage Outbound Message Type (D4-HNLBATUSG)
- **Description**: NES Handle Batch Usage
- **Business Object**: D1-OutboundMessage
- **Priority**: Priority 20 (or lower)

External Systems

Add the event and usage extract Outbound Message Types and message senders to the external system used for the NES head-end system as follows:

- **External System**: NES
- **Description**: NES
- **Outbound Message Types**:

Outbound Message Type	Message Sender
D4-HNLBATEVT	D4-HNLBATEVT
D4-HNLBATUSG	D4-HNLBATUSG

Note: The following apply to the above outbound message types:

- **Processing Method**: Real-time
- **Message XSL**: D4-Request xsl
- **Response XSL**: D4-Response.xsl

Activity Type - NES Event Extract Scheduler

Create an Activity Type for event extract requests as follows:

- **Activity Type**: Enter a code for the activity type (NES-EVENT-EXTRACT)
- **Description**: Enter a description for the activity type (NES Event Extract Scheduler Type)
- **Related Transaction BO**: NES Event Extract Scheduler
- **Activity Type Status**: Active
- **Outbound Message Type**: NES Handle Batch Event
- **External System**: The external system used for the NES head-end system

- **Exception Handling**:
  - **To Do Type**: Activity To Do Type
  - **To Do Role**: System Default Role
  - **Retry Frequency**: Select retry frequency in case if Activity reaches an error state (00:03:00)
  - **Maximum Retries**: Select max number of times the activity will be retried - since it reached the error state (1)
  - **Discard Reason**: Select from the dropdown if available.

Activity Type - NES Usage Extract Scheduler

Create an Activity Type for usage extract requests as follows:

- **Activity Type**: Enter a code for the activity type (NES-USAGE-EXTRACT)
- **Description**: Enter a description for the activity type (NES Usage Extract Scheduler Type)
- **Related Transaction BO**: NES Usage Extract Scheduler
- **Activity Type Status**: Active
- **Outbound Message Type**: NES Handle Batch Usage
- **External System**: The external system used for the NES head-end system
- **Exception Handling**:
  - **To Do Type**: Activity To Do Type
  - **To Do Role**: System Default Role
  - **Retry Frequency**: Select retry frequency in case if Activity reaches an error state (00:03:00)
  - **Maximum Retries**: Select max number of times the activity will be retried - since it reached the error state (1)
  - **Discard Reason**: Select from the dropdown if available.

Activity - NES Event Extract Scheduler

Initiate an Activity for event extract requests as follows:

1. Select **Main**->**Communication**->**Activity+**
2. Select "NES Event Extract Scheduler Type" from the **Activity Type** drop-down list.
3. Enter the number of hours for each extract request in the **Number of Hours of Data Request** field. The default is 24.
4. *Optional*: Enter start and end dates and times for the first extract request in the **Current Request Start Date/Time** and **Current Request End Date/Time** fields.

   Note that these are optional fields. If not populated, the values are populated as follows:

   - **Current Request Start Date/Time**: taken from the last request stop date/time (which is populated by the “Event Extract Scheduler Send Request” algorithm (D4-EVTSNDREQ)
   - **Current Request End Date/Time**: the Current Request Start Date/Time plus the number of hours in the **Number of Hours of Data Request** field.

Activity - NES Usage Extract Scheduler

Initiate an Activity for usage extract requests as follows:

1. Select **Main**->**Communication**->**Activity+**
2. Select "NES Usage Extract Scheduler Type" from the **Activity Type** drop-down list.
3. Enter the number of hours for each extract request in the **Number of Hours of Data Request** field. The default is 24.

4. *Optional*: Enter start and end dates and times for the first extract request in the **Current Request Start Date/Time** and **Current Request End Date/Time** fields.

   Note that these are optional fields. If not populated, the values are populated as follows:

   - **Current Request Start Date/Time**: taken from the last request stop date/time (which is populated by the "Usage Extract Scheduler Send Request" algorithm (D4-USGSNDREQ)
   - **Current Request End Date/Time**: the Current Request Start Date/Time plus the number of hours in the **Number of Hours of Data Request** field.

Executing Extract Requests

To schedule and execute usage and device event extract requests, set up the "Usage / Event Extract Scheduler Monitor" batch process (D1-EXTSC) to run at a frequency appropriate to when you wish to retrieve usage and device events from the NES. For example, to retrieve usage and device events on a daily basis, schedule the "Usage / Event Extract Scheduler Monitor" batch process (D1-EXTSC) to run once every 24 hours.

Sensus

The Oracle Utilities Smart Grid Gateway Adapter for Sensus RNI supports communication with the Sensus Regional Network Interface (RNI), including measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, status check, and on-demand read. The following table describes the attributes of the adapter:

Attribute	Details
Currently Supported Version	Sensus RNI 4.4
Protocol	MultiSpeak 3.0 & 4.1. RNI 3.1 and the SGG Sensus RNI adapter support MultiSpeak 4.1
Market(s)	Worldwide
Architecture	Long range radio WAN (mesh)

Sensus Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, and Oracle Utilities Application Framework (OUAF) objects that are supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data and Device Event Loading

The initial measurement data load and device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway (SGG) service format. Payloads contain measurements and meter events in some head-end specific format. OSB then places each service call into a Java Message Service (JMS) queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel. A service then creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format.

Initial Measurements

The usage data exported from the AMI head-end system as a file in Sensus RNI format is loaded into Oracle Utilities as Initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. **SGG-D6-USAGE-BASE** contains components responsible for “actual” processing of incoming data. It should not be modified during configuration. This can be upgraded without affecting the customization and environment settings added to SGG-D6-USAGE-CM.
2. **SGG-D6-USAGE-CM** allows for customization and simplifies future upgrades.

The runtime configuration settings for the SGG-D6-USAGE-CM project are stored in the xquery file EnvironmentSettings.xq. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRawIMD	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	
destinationRootElementScalar	Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval.	
dateTimeInUTC	Indicates whether the Sensus RNI system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device.	true false
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterUsage	Determines if usage should be filtered.	true false
useExternalTOU	Indicates whether or not an externally reference TOU period should be used when processing measurement data.	true false

Publishing Initial Measurement Data

The SGG adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Initial Measurement Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D6-USAGE-CM OSB project are used in publishing measurement data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Initial Measurement Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Initial measurement data is published in the "native" initial measurement data format (the format of the initial measurement seeder business object). This format includes normalized unit of measure and condition codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published through this feature is published prior to validation, estimation, and editing (VEE) processing. In addition, filtering *cannot* be applied to data published through this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Initial Measurement Data

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to "true" as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **Sensus UOM Code to Standard UOM Mapping** extendable lookup (D6-HeadendUOMLookup) are passed into the system for processing.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service for the project. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Prioritized Initial Measurement Processing

The SGG adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to “Real Time” (D1RT). In addition, initial measurements received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Device Events

The device event data exported from the head-end system as a file in Sensus RNI format is loaded into Oracle Utilities as a Device Event. One of your configuration tasks is to customize the device events processing.

The required functionality is delivered in the base product as two OSB projects:

1. **SGG-D6-EVENT-BASE** contains components responsible for “actual” processing of incoming data. It can be upgraded in future without affecting the customization and environment settings that done in SGG-D6-EVENT-CM project.
2. **SGG-D6-EVENT-CM** allows the customization and simplifies the future upgrades.

The runtime configuration settings for the SGG-D6-EVENT-CM project are stored in the xquery file EnvironmentSettings.xq. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the event data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElement	Holds the name of inbound web service for the device event seeder.	
dateTimeInUTC	Indicates whether the Sensus RNI system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device.	true false
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false

Publishing Events

The Sensus adapter can be configured to publish device events for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Device Event Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D6-EVENT-CM OSB project are used in publishing device events data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Device Event Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Device event data is published in the "native" device event data format (the format of the device event seeder business object). This format includes normalized device event codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: In addition, filtering can NOT be applied to device events published via this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging

Parameter	Description	Default Value
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Events

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **Sensus Device Event Mapping** extendable lookup (D6-DeviceEventMappingLookup) are passed into the system for processing.

Prioritized Device Event Processing

The SGG adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Configuring Payload Processing Notifications

By default, Payload Statistics, Payload Error Notification, and Payload Summary notification activities are sent to the Notification Queue within Oracle Service Bus. Smart Grid Gateway adapters can be configured to restrict sending these activities by defining optional JCA transport dynamic properties in the appropriate OSB project. The following optional JCA transport dynamic properties can used to control if these notification activities are sent:

- **SGG_SEND_STATISTICS**: Controls sending Payload Statistic activities (true: send, false: do not send)
- **SGG_SEND_ERROR**: Controls sending Payload Error Notification activities (true: send, false: do not send)
- **SGG_SEND_SUMMARY**: Controls sending Payload Summary activities (true: send, false: do not send)

When these properties are set to false, the corresponding payload notification will not be sent to the JMS Queue. Note these properties are not populated by default, and default to true if omitted.

To create these properties:

1. Navigate to the SGG-D6-EVENT-CM project in the Oracle Service Bus Console.
2. Expand the **Proxy Services** folder.
3. Select the **InboundProxyService** proxy service (not the Pipeline or WSDL).
4. Click **Create** to create a new session.
5. Select **Transport Details**.
6. Click the plus sign in the **Dyanmic EndPoint Properties** section. An empty row will appear.
7. Enter the property you wish to define in the **Property** column.
8. Enter "false" in the **value** column.
9. Repeat steps 6-8 for each property you wish to define.
10. Click **Activate** to save and activate your changes.

NOTE: These properties are only used if the SGG_EXT_ID_PATTERN is not present or the regular expression captures nothing. The SGG_EXT_ID_PATTERN will override these properties when the regular expression captures group data.

Base Package Business Objects

The Sensus RNI adapter base package includes the following device and initial measurement business objects:

Business Object Name	Description
D6-InitialLoadIMDInterval	Sensus Initial Load IMD - Interval Used when loading Sensus interval measurements into the system for the first time.
D6-InitialLoadIMDScalar	Sensus Initial Load IMD - Scalar
D6-SmartMeter	Sensus Smart Meter

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, an on-demand read, or another type of request that the head-end system supports. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related web service from the head-end system. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

Communication Flows

The table below lists the communications created for each Sensus RNI command:

Command	Outbound Communication	Inbound Communication	Completion Event
Remote Connect	Sensus – Initiate Connect Disconnect	Sensus – Connect / Disconnect State Change	Connect Device Completion Event
Remote Disconnect	Sensus – Initiate Connect Disconnect	Sensus – Connect / Disconnect State Change	Disconnect Device Completion Event
Device Commissioning	Sensus – Meter Add Notification		Device Commissioning Completion Event
Device Decommissioning	Sensus – Meter Remote Notification		Device Decommissioning Completion Event
On-Demand Read (Scalar)	Sensus – Initiate Meter Read By Meter ID	Sensus – Reading Changed Notification	Create IMD Completion Event
On-Demand Read (Interval)	Interval data not supported		
Device Status Check	Sensus – Initiate Outage Detection	Sensus – Outage Detection Event Notification	

Device Communication Base Package Business Objects

The Sensus RNI Adapter base package includes the following communication business objects:

Business Object Name	Description
D6-ConnectDisconStateChgNtf	Sensus - Connect/Disconnect State Change
D6-InitiateConnectDisconnect	Sensus - Initiate Connect Disconnect

Business Object Name	Description
D6-InitiateMeterByMeterId	Sensus - Initiate Meter Read By Meter ID
D6-InitiateOutageDetection	Sensus - Initiate Outage Detection
D6-MeterAddNotification	Sensus - Meter Add Notification
D6-MeterRemoveNotifcation	Sensus - Meter Remove Notification
D6-OutageDetectEvtNotification	Sensus - Outage Detection Event Notification
D6-ReadingChgNotification	Sensus - Reading Changed Notification
D6-UnsolicitedEvtNotification	Sensus - Unsolicited Event Notification

Sensus Event Data Mapping

The Sensus event file format maps as follows into the business object, D1-DeviceEventMappingLookup:

Sensus Flat File Field	Device Event Seeder BO Element	Comments
Transaction ID (from Header record)	External Source Identifier	This is the file name.
Device Identifier	External Device Identifier	
Event Name	External Event Name	
Event Creation Date/Time	Event Date/Time	
Device Type	External Device Type	This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form): Meter Collector Router
Service Location ID	External Service Location ID	
Communication Module Serial Number	External Communication Module Identifier	
Event Category ID	External Event Category	
Event Severity	External Event Severity	Valid values include (although the element itself is free-form): Alert Information
Status Value	External Status Value	This represents additional information that relates to the event itself.
Status Date/Time	External Status Date/Time	The date & time at which the additional information referenced above had occurred.

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch or Real-time)
- The corresponding message senders

- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgement and response messages are sent and received validating that commands have been transmitted.

Outbound Message Type	Description
D6-CONDISCON	Sensus Initiate Connect Disconnect
D6-INITMTR	Initiate Meter Read By Meter ID
D6-INTOUTDET	Initiate Outage Detection Request

Inbound / Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The Oracle Utilities Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway adapter for Sensus RNI includes the following inbound web services:

Inbound Web Service	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Used for upload of device events. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-InitialLoadIMD	Used for initial measurement upload. The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.
D1-DeviceStatusCheck	Device Status Check This service is invoked by the integration layer to instantiate a Device Status Check command.
D1-InitialLoadIMD	Used by OSB to instantiate an IMD This inbound web service is used by OSB to instantiate an Initial Measurement Data for incoming interval usage in the Sensus format.

Inbound Web Service	Description
D1-RemoteConnect	Remote Connect This service is invoked by the integration layer to instantiate a Remote Connect command.
D1-RemoteDisconnect	Remote Disconnect This service is invoked by the integration layer to instantiate a Remote Disconnect command.
D6-ConDisconStChgNotification	Initiate Connect Disconnect response. Retrieve response from the Initiate Connect Disconnect command.
D6-OutageDetectionEventNotification	Initiate Outage Detection Response Retrieve response from the Initiate Outage Detection Event Notification command.
D6-ReadingChangedNotification	Reading Changed Notification Notification that a Sensus device reading has changed.
D6-UnsolicitedEventNotification	Unsolicited Event Response Retrieve unsolicited notifications when an event triggers an alarm on the meter.

Message Senders

Message senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of message senders you need to create is based on the types of messages the system is designed to accept.

The Oracle Utilities Smart Grid Gateway adapter for Sensus RNI includes the following message senders:

Message Sender	Description
D6-CONDISCON	Sensus Initiate Connect/Disconnect
D6-INTOUTDET	Initiate Outage Detection Request
D6-InitMID	Initiate Meter Read By Meter ID Outbound Message

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to MultiSpeak 4.1 format, invoking process callouts and invoking the remote endpoint to trigger the device events.

OnDemandRead Composite Process — Invokes the remote endpoint to trigger the on-demand read event. An asynchronous reply responds to the OUAF layer when the reading arrives.

ConnectDisconnect Composite Process — Invokes the remote endpoint to trigger the connect/disconnect event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

CommissionDecommission Composite Process — Invokes the remote endpoint to trigger the commission or decommission event. After the synchronous call completes, a one of the following second business callout services is invoked to determined if the related “received” or “completed” callout should be executed:

- isExecutingCommissionReceivedCallout
- isExecutingCommissionCompletedCallout
- isExecutingDecommissionReceivedCallout
- isExecutingDecommissionCompletedCallout

DeviceStatusCheck Composite — Invokes the remote endpoint to trigger the initiate outage detection event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

ProcessCallout Composite — This business callout provides a point at which customers and implementers can incorporate custom business logic and transformations. This composite includes the WSDLs and processing logic for all of the MultiSpeak processes. The default implementation of each method is a direct return of the input.

Web Services

These web services are all defined in the Sensus RNI head end system. The WSDLs were added to a Meta Data Storage (MDS) layer in OUAF and all references to the WSDL point to this MDS location. These web services have HTTP security by default. You may need to modify the security as a part of your implementation.

Web Service	Related BPEL Process	Description
CB_ServerService	ConnectDisconnect	This web service defines the return interface, the means by which the status is returned to the calling system. This web service is only be invoked by the head end system, not OUAF. Only the CDStateChangeNotification web method is implemented in the composite. The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Sensus/ConnectDisconnect/CB_ServerService
CB_Server	OnDemandRead	This web service defines the return interface, the means by which the reading is returned to the calling system. This web service is only be invoked by the head end system, not OUAF. Only the ReadingChangedNotification web method is implemented in the composite. The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Sensus/OnDemandRead/CB_Server
OA_ServerService	DeviceStatusCheck	This web service defines the asynchronous return for InitateOutageDetectionEventRequest for solicited responses. It is also used for unsolicited alarms. This web service is only be invoked by the head end system, not OUAF. Only the ODEventNotification, PingURL, and GetMethods web methods are implemented in the composite. The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Sensus/DeviceStatusCheck/OA_ServerService

Sensus RNI Web Services

The following table describes the Sensus RNI web services and operations used for the Oracle Utilities Smart Grid Gateway command messaging:

Smart Grid Gateway Command	AMI Adapter Business Objects	Sensus Web Services	Sensus Operations
Device Commissioning	D6-MeterAddNotification	MR	MeterAddNotification
Device Decommissioning	D6-MeterRemoveNotification	MR	MeterRemoveNotification

Smart Grid Gateway Command	AMI Adapter Business Objects	Sensus Web Services	Sensus Operations
Remote Connect/ Remote Disconnect	D6-InitiateConnectDisconnect	CD	InitiateConnectDisconnect
	D6-ConnectDisconStateChgNtf	CB	CDStatesChangedNotification (async reply)
Device Status Check	D6-InitiateOutageDetection	OD	InitiateOutageDetectionEventRequest
	D6-OutageDetectEvtNotification	OA	ODEventNotification (async reply)
On-Demand Read	D6-InitiateMeterByMeterId	MR	InitiateMeterReadingsByMeterID
	D6-ReadingChangedNotification	CB	ReadingChangedNotification (async reply)

Configuring a Sensus Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway Adapter for Sensus RNI to communicate with the Sensus RNI.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the Sensus Regional Network Interface (RNI) in response to a command request.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-DeviceEventSeeder	Device Event Seeder
D1-DeviceStatusCheck	Device Status Check
D1-InitialLoadIMD	IMD Seeder
D1-PayloadErrorNotif	Payload Error Notification
D1-PayloadStatistics	Payload Statistics
D1-PayloadSummary	Payload Summary
D6-ConDisconStChgNotification	Initiate Connect Disconnect Response
D6-OutageDetectionEventNotification	Initiate Outage Detection Response
D6-ReadingChangedNotification	Reading Changed Notification
D6-UnsolicitedEventNotification	Unsolicited Event Response

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object

- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

Message Senders

Message senders define the details of how messages are sent to an external system, such as messages containing device command requests.

The following message senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders.

Message Sender	Description
D6-CONDISCON	Sensus Initiate Connect/Disconnect
D6-INTOUTDET	Initiate Outage Detection Request
D6-InitMID	Initiate Meter Read By Meter ID Outbound Message

Note: The following apply to all of the above message senders:

Main Tab:

- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction:<**OPERATION**>
- **HTTP Login User**: <**USER_ID**>
- **HTTP Login Password**: <**PASSWORD**>
- **HTTP Method**: POST
- **HTTP URL 1**: http://<**EM_SERVER**>:<**EM_SERVER_PORT**>/soa-infra/services/Sensus/<**SERVICE**>
- where:
  - <**OPERATION**>: the operation performed by the message sender (see Operation column in the table above)
  - <**USER_ID**>: the user ID used to log into WebLogic Enterprise Manager
  - <**PASSWORD**>: the password used to log into WebLogic Enterprise Manager
  - <**EM_SERVER_IP**>: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
  - <**EM_SERVER_PORT**>: the port where the WebLogic Enterprise Manager is installed
  - <**SERVICE**>: the service invoked by the message sender (see Service column in the table above)

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
D6-CONDISCON	Sensus Initiate Connect Disconnect
D6-INITMTR	Initiate Meter Read By Meter ID

Outbound Message Type	Description
D6-INTOUTDET	Initiate Outage Detection Request

Note: The following apply to all of the above outbound message types:

- **Business Object**: D1-OutboundMessage (Outbound Message)
- **Priority**: Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the Sensus RNI must be present in your system. If this is not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System - Sensus:

- **External System**: Sensus
- **Description**: Sensus
- **Outbound Message Types**:

Outbound Message Type	Description	Message Sender
D6-CONDISCON	Sensus Initiate Connect Disconnect	D6-CONDISCON
D6-INITMTR	Initiate Meter Read By Meter ID	D6-InitMID
D6-INTOUTDET	Initiate Outage Detection Request	D6- INTOUTDET

Note: The following apply to all of the above outbound message types:

- **Processing Method**: Real-time
- **Message XSL**: D6-Request xsl
- **Response XSL**: D6-Response.xsl

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the Sensus RNI must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Meter Data Management / Smart Grid Gateway documentation for more information about creating service providers.

Service Provider - Sensus:

- **Service Provider**: Sensus
- **Description**: Sensus
- **External Reference ID**: Sensus
- **External System**: Sensus
- **Our Name/ID in Their System**:

- **AMI Device ID Type**: Internal Meter Number
- **AMI Measuring Component ID Type**: Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to a given head-end system.

The following types of processing methods must be configured for the Sensus service provider. Refer to the Oracle Utilities Meter Data Management / Smart Grid Gateway documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head-end system.

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.

UOM Translation

UOM mapping processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the Sensus service provider, based on the requirements of each implementation using the “How to Create OB COMM/Send OB Message” processing method business object (D1–HowToCreateActivityOBComm).

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Registration	D6-MeterAddNotification	Commission
Device Decommission	Device Removal	D6-MeterRemoveNotification	Decommission
Device Status Check	Device Status Check	D6-InitiateOutageDetection	Initiate Outage Detection Request
On-Demand Read (Scalar)	On-Demand Read (Scalar)	D6-InitiateMeterByMeterId	Initiate Meter Read by Meter ID
Remote Connect	Remote Connect	D6-InitiateConnectDisconnect	Connect Device
Remote Disconnect	Remote Disconnect	D6-InitiateConnectDisconnect	Disconnect Device

Configuring Endpoint URIs

Part of the configuration process is configuring your BPEL composites to work with your head-end system by defining the appropriate Endpoint URIs for each of the commands. The default approach to defining Endpoints URIs is redeployment or reinstallation or the BPEL composites. For example, by default, changing an adapter from using the test harness to a production environment using the actual head-end system requires editing the appropriate installation menu options and redeploying the BPEL composites. The Endpoints URIs defined during installation and deployment for each adapter are listed in the **Smart Grid Gateway Installation and Configuration Worksheets** section of the *Oracle Utilities Smart Grid Gateway Installation Guide*.

You can also use an "Endpoint Override" Domain Value Map (DVM) to override Endpoint URIs defined during deployment and installation. This DVM allows defining specific keys that provide an alternate URL that will override the original installed value. Endpoint Override DVMs can be edited using the Oracle SOA Composer.

The Sensus endpoint override DVM (D6–EndpointOverrides.dvm) uses a specific set of keys, each used for one or more commands. The table below lists the DVM keys available for the Sensus adapter and the command used with each:

DVM Key	Commands
MR_Server	Device Commissioning
	Device Decommissioning
	On-Demand Read (Scalar)
CD_Server	Remote Connect
	Remote Disconnect
OD_Server	Device Status Check
ProcessCallout	User Exit Functions

To define an override Endpoint URI for the Sensus adapter, use the following procedure:

1. Open the SOA Composer for your BPEL configuration.

   The URL for the SOA Composer is http://server:port/soa/composer.

2. Select the D6–EndpointOverrides.dvm in the left panel.

   - In Fusion Middleware v12.2.1, this is located under **Shared** in the **Deployment View**, or under **Domain Value Maps** in the **Types View**.
   - In Fusion Middleware v12.2.2, this is located under **Metadata** in the **Deployment View**.

   The DVM will open in the right panel. The **Description** field lists the available keys for the DVM (only the first key is shown, but you can scroll through the contents to view the list. The panel also displays a list of previously defined keys.

3. Click the **Create Session** button (above the left panel) to begin an editing session.

4. Click the **Add Domain Values** icon ("+") to add a new key.

   The **Add Domain Values** dialog opens.

5. Enter the appropriate values in the **Add Domain Values** dialog as follows:

   - **key**: The DVM key for the Endpoint URI you wish to define (see the table above).
   - **EndpointURI**: The override Endpoint URI.

6. Click **OK**.

   The new DVM value will appear in the list of keys.

7. Click the **Save** icon to save the DVM values.

8. Click the **Publish** button to activate all the changes in the editing session.

   Enter an optional note for the session in the **Publish Session** dialog.

Changes take effect immediately upon publishing the session.

Click **Discard** to discard your changes.

Click **Exit** to exit your current session. Note that your session will still be open if you exit. Use **Discard** to end your session without making changes.

Configuring Sensus Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Sensus.

This section outlines some of the extendable lookups that must be configured for use with the Sensus adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system. Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name**: The event name used by the head-end system
- **Description**: A description of the device event
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Standard Event Name**: The standard event name for device events of this type, from the “Standard Event Name” extendable lookup.

UOM Code to Standard UOM Mapping

Usage received from a utility’s head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The UOM Code to Standard UOM Mapping extendable lookup is used for this purpose. Each value defined for the UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-end UOM**: The unit of measure code used by the head-end system.
- **Unit of Measure**: The unit of measure defined in the system.
- **Description**: A description of the unit of measure code.
- **Status**: The status of the lookup value (can be Active or Inactive)

Interval Status Code to Condition Mapping

Interval usage received from the Sensus RNI can include Sensus interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The Sensus Interval Status Code to Condition Mapping extendable lookup is used to determine how to map Sensus interval status codes to standard status codes when receiving usage from the Sensus RNI.

Each value defined for the Sensus Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status**: The Sensus interval status code
- **Description**: A description of the interval status code.
- **Status**: The status of the lookup value (can be Active or Inactive)

- **Condition**: The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.

Other Extendable Lookups

Business Object Name	Description
D6-CDReasonCodeLookup	Sensus Connect/Disconnect Reason Code
D6-LoadActionCodeLookup	Sensus Load Action Code
D6-OutageEventTypeLookup	Sensus Outage Event Type
D6-SensusTimeUnits	Sensus Time Units
D6-ServiceTypeMappingLookup	Sensus Service Type Mapping

Using the Sensus Test Harness

Oracle Utilities Smart Grid Gateway Adapter for Sensus RNI includes a test harness that can be configured to simulate a general head-end system for testing the two-way commands. The test harness includes a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. See Using Smart Grid Gateway Test Harnesses for more information.

Silver Spring Networks

The Oracle Utilities Smart Grid Gateway Adapter for Silver Spring Networks supports communication with the Silver Spring Networks UtilityIQ application, including measurement data and device event loading, and command messaging in support of commissioning, connect, disconnect, decommissioning, status check, and on-demand read. The following table describes the attributes of the adapter:

Attribute	Details
Currently Supported Versions	UtilityIQ Version 4.10
Protocol	Proprietary
Market(s)	Worldwide
Architecture	RF WAN (mesh) based on Access Points

Silver Spring Networks Adapter Processing

This section provides details concerning the OSB processing, BPEL Processes, and Oracle Utilities Application Framework (OUAF) objects that are supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data and Device Event Loading

The initial measurement data load and device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway (SGG) service format. Payloads contain measurements and meter events in some head-end specific format. OSB then places each service call into a Java Message Service (JMS) queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel. A service then creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format.

Initial Measurements

The usage data exported from the AMI head-end system as a file in Silver Spring Networks XML format is loaded into Oracle Utilities as initial measurement data. The following OSB projects, delivered in the base product, help manage the usage processing:

1. SGG-D7-SSNXML-BASE contains components responsible for “actual” processing of incoming data. It should not be modified during configuration. This project can be upgraded without affecting the customization and environment settings added to SGG-D7-SSNXML-CM.
2. SGG-D7-SSNXML-CM allows for customization and simplifies future upgrades.

The runtime configuration settings for the SGG-D7-SSNXML-CM project are stored in the xquery file EnvironmentSettings.xq. You can use this file to adjust initial measurement data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRawIMD element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the initial measurement data is populated as raw data. Valid values are:	true false
callPreProcessing	Determines if the preprocessing proxy service is called. Valid values are:	true false
callPostProcessing	Determines if the postprocessing proxy service is called. Valid values are:	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	
destinationRootElementScalar	Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval.	
destinationRootElementEvent	Holds the name of inbound web service for the device event seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false
filterUsage	Determines if usage should be filtered.	true false

Publishing Initial Measurement Data

The SGG adapter can be configured to publish initial measurement data for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Initial Measurement Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D7-SSNXML-CM OSB project are used in publishing measurement data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Initial Measurement Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Initial measurement data is published in the "native" initial measurement data format (the format of the initial measurement seeder business object). This format includes normalized unit of measure and condition codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published through this feature is published prior to validation, estimation, and editing (VEE) processing. In addition, filtering *cannot* be applied to data published through this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Initial Measurement Data

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to "true" as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **SSN - UOM Code to Standard UOM Mapping** extendable lookup (D7-HeadendUOMLookup) are passed into the system for processing.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the "InboundProxyService" proxy service for the project. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Prioritized Initial Measurement Processing

The SGG adapter prioritizes processing of initial measurements created from smart meter commands and/or completion events by setting the Execution Method flag in these types of initial measurements to "Real Time" (D1RT). In addition, initial measurements received with the Execution Method flag set to "Real Time" will be processed in real time rather than via batch processing. See **Initial Measurement Data Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Device Events

The device event data exported from the head-end system as a file in Silver Spring Networks CSV format is loaded into Oracle Utilities as a device event. One of your configuration tasks is to customize the device events processing.

The required functionality is delivered in the base product as two OSB projects:

1. SGG-D7-CSV-BASE contains components responsible for "actual" processing of incoming data. It can be upgraded in future without affecting the customization and environment settings that done in SGG-D7-CSV-CM project.
2. SGG-D7-CSV-CM allows the customization and simplifies the future upgrades.

The runtime configuration settings for the SGG-D7-CSV-CM project are stored in the EnvironmentSettings.xq XQuery file. You can use this file to adjust device event data processing. For example, if you want to load raw data you would specify "true" for the content of the populateRaw element.

The following table describes the elements included in the EnvironmentSettings.xq file:

Element	Description	Valie Values
populateRaw	Determines if the device event data is populated as raw data. Valid values are:	true false
callPreProcessing	Determines if the preprocessing proxy service is called. Valid values are:	true false
callPostProcessing	Determines if the postprocessing proxy service is called. Valid values are:	true false
destinationRootElement	Holds the name of inbound web service for the device event seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered. Valid values are:	true false

Publishing Events

The Silver Spring Networks adapter can be configured to publish device events for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Device Event Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-D7-CSV-CM OSB project are used in publishing device events data to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Device Event Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems. Device event data is published in the “native” device event data format (the format of the device event seeder business object). This format includes normalized device event codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: In addition, filtering can NOT be applied to device events published via this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging

Parameter	Description	Default Value
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Events

The SGG adapter can be configured to filter initial measurement data passed into SGG and the Oracle Utilities Meter Data Management application. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **SSN - Device Event Mapping** extendable lookup (D7-DeviceEventMappingLookup) are passed into the system for processing.

Prioritized Device Event Processing

The SGG adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Configuring Payload Processing Notifications

By default, Payload Statistics, Payload Error Notification, and Payload Summary notification activities are sent to the Notification Queue within Oracle Service Bus. Smart Grid Gateway adapters can be configured to restrict sending these activities by defining optional JCA transport dynamic properties in the appropriate OSB project. The following optional JCA transport dynamic properties can used to control if these notification activities are sent:

- **SGG_SEND_STATISTICS**: Controls sending Payload Statistic activities (true: send, false: do not send)
- **SGG_SEND_ERROR**: Controls sending Payload Error Notification activities (true: send, false: do not send)
- **SGG_SEND_SUMMARY**: Controls sending Payload Summary activities (true: send, false: do not send)

When these properties are set to false, the corresponding payload notification will not be sent to the JMS Queue. Note these properties are not populated by default, and default to true if omitted.

To create these properties:

1. Navigate to the SGG-D7-CSV-CM project in the Oracle Service Bus Console.
2. Expand the **Proxy Services** folder.
3. Select the **InboundProxyService** proxy service (not the Pipeline or WSDL).
4. Click **Create** to create a new session.
5. Select **Transport Details**.
6. Click the plus sign in the **Dyanmic EndPoint Properties** section. An empty row will appear.
7. Enter the property you wish to define in the **Property** column.
8. Enter "false" in the **value** column.
9. Repeat steps 6-8 for each property you wish to define.
10. Click **Activate** to save and activate your changes.

NOTE: These properties are only used if the SGG_EXT_ID_PATTERN is not present or the regular expression captures nothing. The SGG_EXT_ID_PATTERN will override these properties when the regular expression captures group data.

Base Package Business Objects

The Silver Spring Networks adapter base package includes the following device and initial measurement business objects:

Business Object Name	Description
D7-InitialLoadIMDInterval	SSN - Initial Load IMD - Interval Used when loading Silver Spring Network (SSN) interval measurements into the system for the first time.
D7-InitialLoadIMDScalar	SSN - Initial Load IMD - Scalar Used when loading Silver Spring Network (SSN) scalar measurements into the system for the first time.
D7-SmartMeter	SSN — Smart Meter

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, an on-demand read, or another type of request that the head-end system supports. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related web service from the head-end system. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

Communication Flows

The table below lists the communications created for each Silver Spring Networks command:

Command	Outbound Communication	Inbound Communication	Completion Event
Remote Connect	SSN–Connect or Disconnect	SSN – Connect or Disconnect Response Remote Provisioning Job (Get Status)	Connect Device Completion Event
Remote Disconnect	SSN–Connect or Disconnect	SSN – Connect or Disconnect Response	Disconnect Device Completion Event
Device Commissioning	SSN- Replace Location		Device Commissioning Completion Event
Device Decommissioning	SSN- Replace Device At Location (Decomm)		Device Decommissioning Completion Event
On-Demand Read (Scalar)	SSN – Add Meter Read Job (Scalar)	SSN – Meter Read Response (Scalar)	Create IMD Completion Event
On-Demand Read (Interval)	SSN – Add Meter Read Job (Interval)	SSN – Meter Read Response (Interval)	Create IMD Completion Event
Device Status Check	SSN – Add Ping job	SSN – Ping Job Response	

Device Communication Base Package Business Objects

The Silver Spring Networks Adapter base package includes the following communication business objects:

Business Object Name	Description
D7-AddMeterReadJobInterval	SSN - Add Meter Read Job (Interval)
D7-AddMeterReadJobScalar	SSN - Add Meter Read Job (Scalar)
D7-AddPingJob	SSN - Add Ping Job
D7-ConnectDisconnect	SSN - Connect or Disconnect
D7-ConnectDisconnectResp	SSN - Connect or Disconnect Response
D7-GetStatus	SSN - Get Status
D7-GetStatusResponse	SSN - Get Status Response
D7-MeterReadResponseInterval	SSN - Meter Read Response (Interval)
D7-MeterReadResponseScalar	SSN - Meter Read Response (Scalar)
D7-PingJobResponse	SSN - Ping Job Response
D7-ReplaceDeviceAtLocForDecomm	SSN - Replace Device At Location (Decomm)
D7-ReplaceLocation	SSN - Replace Location

Silver Spring Networks Event Data Mapping

The Silver Spring event file format maps as follows into the business object, D1-DeviceEventMappingLookup:

Silver Spring Flat File Field	Device Event Seeder BO Element	Comments
Transaction ID (from Header record)	External Source Identifier	This is the file name.
Device Identifier	External Device Identifier	
Event Name	External Event Name	
Event Creation Date/Time	Event Date/Time	
Device Type	External Device Type	This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form): Meter Collector Router
Service Location ID	External Service Location ID	
Communication Module Serial Number	External Communication Module Identifier	
Event Category ID	External Event Category	
Event Severity	External Event Severity	Valid values include (although the element itself is free-form): Alert Information
Status Value	External Status Value	This represents additional information that relates to the event itself.
Status Date/Time	External Status Date/Time	The date & time at which the additional information referenced above had occurred.

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch or Real-time)
- The corresponding message senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Outbound Message Types

Acknowledgment and response messages are sent and received validating that commands have been transmitted.

Outbound Message Type	Description
D7-COMMS	Replace Device At Location
D7-OB MSG TY	Outbound Message Type SSN

Inbound/Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The Oracle Utilities Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound web services

Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway adapter for Silver Spring Networks includes the following inbound web services:

Inbound Web Service	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Used for upload of device events. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-DeviceStatusCheck	Device Status Check This service is invoked by the integration layer to instantiate a Device Status Check command.

Inbound Web Service	Description
D1-InitialLoadIMD	Used by OSB to instantiate an IMD The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.
D1-RemoteConnect	Remote Connect This service is invoked by the integration layer to instantiate a Remote Connect command.
D1-RemoteDisconnect	Remote Disconnect This service is invoked by the integration layer to instantiate a Remote Disconnect command.
D7-ConnectDisconnectResponse	Connect Disconnect Response Retrieves response for Remote provisioning Job Connect or Disconnect commands.
D7-GetStatusResponse	D7-GetStatusResponse Retrieve response from the Get Status command.
D7-MeterReadResponseInterval	SSN - Meter Read Response (Interval)
D7-MeterReadResponseScalar	SSN - Meter Read Response (Scalar)
D7-PingJobResponse	SSN - Ping Response Retrieves response from the Ping Job Response command.

Message Senders

Message senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of message senders you need to create is based on the types of messages the system is designed to accept.

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to Silver Spring Networks format, invoking process callouts and invoking the remote endpoint to trigger the device events.

OnDemandRead Composite Process — Provides access points to edge application and handles data between edge application and head end system. It invokes sequence of web methods to head end system and retrieves meter read and send it back to Edge application.

ConnectDisconnect Composite Process — Performs the conversion from Oracle Utilities format to SSN format, invokes process callouts, and invokes the remote endpoint to trigger the connect event. A second, asynchronous reply will call back into the OUAF layer when the status change is completed at the head-end system. Another asynchronous reply will call back into the OUAF layer to send Meter Read Results.

CommissionDecommission Composite Process — Performs the conversion from Oracle Utilities format to SSN UIQ format, invokes process callouts, and invokes the remote endpoint to trigger the commission or decommission of meter.

DeviceStatusCheck Composite — Performs the conversion from Oracle Utilities format to SSN format, invokes process callouts, and makes a call via a proxy to the head-end system starting the Meter Ping operation. In an ideal scenario, the job status is returned as completed and the results are acquired and sent back to OUAF. If the job takes longer, OUAF will initiate a second request that will poll the head end system for the job status. When the job is completed, the results are returned to OUAF.

Common Composite — Contains two main classes of operations: Proxies and ProcessCallouts. Proxies are simple mediators that forward a web service call to a preset endpoint. No transformations are performed. They are convenient because they allow head end URLs and security to be set in a single composite. ProcessCallouts are points of customization which allow users to modify data and/or initiate some external business process.

BulkRequest Composite — Provides access points to requesting application. It decouples the bulk request into single commands for each meter/device in the request and sends it to edge application for processing.

Web Services

The following web services are all defined in the Silver Spring Networks head-end system:

- **CommissionDecommissionService**
  - **BPEL Process:** CommissionDecommission
  - **Operation:** ReplaceDeviceAtLocation
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/SSN/CommissionDecommission/CommissionDecommissionService
- **CommissionDecommissionService**
  - **BPEL Process:** CommissionDecommission
  - **Operation:** ReplaceLocation
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/SSN/CommissionDecommission/CommissionDecommissionService
- **ConnectDisconnectService**
  - **BPEL Process:** ConnectDisconnect
  - **Operation:** AddRemoteProvisioningJob
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/SSN/ConnectDisconnect/ConnectDisconnectService
- **DeviceStatusCheckService**
  - **BPEL Process:** DeviceStatusCheck
  - **Operation:** AddPingJob
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/SSN/DeviceStatusCheck/DeviceStatusCheckService
- **AddMeterReadJobService**
  - **BPEL Process:** AddMeterRead
  - **Operation:** AddMeterReadJob
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/SSN/OnDemandRead/AddMeterReadJobService
- **OnDemandReadService**
  - **BPEL Process:** OnDemandRead
  - **Operation:** GetJobStatus
  - **Enpoint URL:** http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/SSN/OnDemandRead/OnDemandReadService

Silver Spring Networks Utility IQ Web Services

The following table describes the Silver Spring Networks Utility IQ web services and operations used for the Oracle Utilities Smart Grid Gateway command messaging:

Smart Grid Gateway Command	AMI Adapter Business Objects	Silver Spring Networks Web Services	Silver Spring Networks Operations
Device Commissioning	D7-ReplaceLocation	Device Manager	findDevice
			ReplaceLocation
Device Decommissioning	D7-ReplaceDeviceAtLocForDecomm	Device Manager	findDevice
			ReplaceDeviceAtLocation
Remote Connect/ Remote Disconnect	D7-ConnectDisconnect	Device Manager	findDevice
	D7-GetStatus	Job Manager	addRemoteProvisioningJob
		DeviceResults	getJobStatusForDevice
			getRemoteProvisioningResultsByJobID
			findJob
			getJobStatus
			getMeterReadResultsByJobID
Device Status Check	D7-AddPingJob	Device Manager	findDevice
		Job Manager	addPingJob, getJobStatus
		DeviceResults	getPingResultsByJobID
On-Demand Read	D7-AddMeterReadJobInterval	Device Manager	findDevice
	D7-AddMeterReadJobScalar	Job Manager	addMeterReadJob
		DeviceResults	getJobStatus
			getMeterReadResultsByJobID

Configuring a Silver Spring Networks Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway Adapter for Silver Spring Networks to communicate with the Silver Spring Networks UtilityIQ.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the Silver Spring Networks UtilityIQ application in response to a command request.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response

Inbound Web Service Name	Description
D1-DeviceEventSeeder	Device Event Seeder
D1-DeviceStatusCheck	Device Status Check
D1-InitialLoadIMD	Used by OSB to instantiate an IMD
D1-PayloadErrorNotif	Payload Error Notification
D1-PayloadStatistics	Payload Statistics
D1-PayloadSummary	Payload Summary
D7-ConnectDisconnectResponse	Connect Disconnect Response
D7-GetStatusResponse	Get Status Response
D7-MeterReadResponseInterval	SSN - Meter Read Response (Interval)
D7-MeterReadResponseScalar	SSN - Meter Read Response (Scalar)
D7-PingJobResponse	SSN - Ping Response

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object
- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

Message Senders

Message senders define the details of how messages are sent to an external system, such as messages containing device command requests. An message sender should be configured for each command.

The following message senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders.

Message Sender	Description
D7-DECOMM	SSN Replace Device at Location for Decommission
D7-COMM	SSN Replace Location - Commission
D7-ADDJOB	SSN Add Meter Read Job
D7-ADDPING	SSN Add Ping Job
D7-CONNECT	SSN Connect Device
D7-GTSTATUS	SSN Get Status

Note: The following apply to all of the above message senders:

Main Tab:

- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction:http://xmlns.oracle.com/ouaf/ssn/<**OPERATION**>

- **HTTP Login User**: **<USER_ID>**
- **HTTP Login Password**: <**PASSWORD**>
- **HTTP Method**: POST
- **HTTP URL 1**: http://**<EM_SERVER>**:**<EM_SERVER_PORT>**/soa-infra/services/SSN/<**SERVICE**>
- where:
- **<OPERATION>**: the operation performed by the message sender (see Operation column in the table above)
- **<USER_ID>**: the user ID used to log into WebLogic Enterprise Manager
- **<PASSWORD>**: the password used to log into WebLogic Enterprise Manager
- **<EM_SERVER_IP>**: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
- **<EM_SERVER_PORT>**: the port where the WebLogic Enterprise Manager is installed
- <**SERVICE**>: the service invoked by the message sender (see Service column in the table above)

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
D7-COMMS	Replace Device At Location
D7-OB MSG TY	Outbound Message Type SSN

Note: The following apply to all of the above outbound message types:

- **Business Object**: D1-OutboundMessage (Outbound Message)
- **Priority**: Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the Silver Spring Networks UtilityIQ must be present in your system. If this is not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System - Silver Spring Networks:

- **External System**: Silver Spring Networks
- **Description**: Silver Spring Networks
- **Outbound Message Types**:

Outbound Message Type	Description
D7-OB MSG TY	Outbound Message Type SSN

Note: The following apply to all of the above outbound message types:

- **Processing Method**: Real-time

- **Message XSL**: D7-Request xsl
- **Response XSL**: D7-Response.xsl

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the Silver Spring Networks UtilityIQ must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about creating service providers.

Service Provider - Silver Spring Networks:

- **Service Provider**: Silver Spring Networks
- **Description**: Silver Spring Networks
- **External Reference ID**: Silver Spring Networks
- **External System**: Silver Spring Networks
- **Our Name/ID in Their System**:
- **AMI Device ID Type**: Internal Meter Number
- **Pre-Commissioning Device ID Type**:
- **AMI Measuring Component ID Type**: Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including as bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to a given head-end system.

The following types of processing methods must be configured for the Silver Spring Networks service provider. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head-end system.

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the Silver Spring Networks service provider, based on the requirements of each implementation using the "How to Create OB COMM/Send OB Message" processing method business object (D1–HowToCreateActivityOBComm).

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Registration	D7-ReplaceLocation	Commission
Device Decommission	Device Removal	D7-ReplaceDeviceAtLocForDecomm	Decommission
Device Status Check	Device Status Check	D7-AddPingJob	Get Status
On-Demand Read (Scalar)	On-Demand Read (Scalar)	D7-AddMeterReadJobScalar	Read Meter Data
On-Demand Read (Interval)	On-Demand Read (Interval)	D7-AddMeterReadJobInterval	Read Meter Data
Remote Connect	Remote Connect	D7-ConnectDisconnect	Connect Device
Remote Disconnect	Remote Disconnect	D7-ConnectDisconnect	Disconnect Device

Configuring Endpoint URIs

Part of the configuration process is configuring your BPEL composites to work with your head-end system by defining the appropriate Endpoint URIs for each of the commands. The default approach to defining Endpoints URIs is redeployment or reinstallation or the BPEL composites. For example, by default, changing an adapter from using the test harness to a production environment using the actual head-end system requires editing the appropriate installation menu options and redeploying the BPEL composites. The Endpoints URIs defined during installation and deployment for each adapter are listed in the **Smart Grid Gateway Installation and Configuration Worksheets** section of the *Oracle Utilities Smart Grid Gateway Installation Guide*.

You can also use an "Endpoint Override" Domain Value Map (DVM) to override Endpoint URIs defined during deployment and installation. This DVM allows defining specific keys that provide an alternate URL that will override the original installed value. Endpoint Override DVMs can be edited using the Oracle SOA Composer.

The Silver Springs Network endpoint override DVM (D7–EndpointOverrides.dvm) uses a specific set of keys, each used for one or more commands. The table below lists the DVM keys available for the Silver Springs Network adapter and the command used with each:

DVM Key	Commands
DeviceResults4.4	Remote Connect
DeviceResults4.7	Remote Disconnect
DeviceResults4.10	On-Demand Read (Scalar)
	On-Demand Read (Interval)
	Device Status Check
DataAggregation4.4	Not used in this version.
DataAggregation4.7	
DataAggregation4.10	
DeviceManager4.4	Device Commissioning
DeviceManager4.7	Device Decommissioning

DVM Key	Commands
DeviceManager4.10	Remote Connect
	Remote Disconnect
	On-Demand Read (Scalar)
	On-Demand Read (Interval)
	Device Status Check
JobManager4.4	Remote Connect
JobManager4.7	Remote Disconnect
JobManager4.10	On-Demand Read (Scalar)
	On-Demand Read (Interval)
	Device Status Check
ProcessCallout	User Exit Functions

NOTE:
The numbers in the keys above (4.4, 4.7, and 4.10) designate the version of the Silver Springs Network IQ head-end system.

Only a single version of each key should be defined, based on the version of the Silver Springs Network IQ head-end system

To define an override Endpoint URI for the Silver Springs Network adapter, use the following procedure:

1. Open the SOA Composer for your BPEL configuration.

   The URL for the SOA Composer is http://server:port/soa/composer.

2. Select the D7–EndpointOverrides.dvm in the left panel.

   - In Fusion Middleware v12.2.1, this is located under **Shared** in the **Deployment View**, or under **Domain Value Maps** in the **Types View**.
   - In Fusion Middleware v12.2.2, this is located under **Metadata** in the **Deployment View**.

   The DVM will open in the right panel. The **Description** field lists the available keys for the DVM (only the first key is shown, but you can scroll through the contents to view the list. The panel also displays a list of previously defined keys.

3. Click the **Create Session** button (above the left panel) to begin an editing session.

4. Click the **Add Domain Values** icon (“+”) to add a new key.

   The **Add Domain Values** dialog opens.

5. Enter the appropriate values in the **Add Domain Values** dialog as follows:

   - **key**: The DVM key for the Endpoint URI you wish to define (see the table above).
   - **EndpointURI**: The override Endpoint URI.

6. Click **OK**.

   The new DVM value will appear in the list of keys.

7. Click the **Save** icon to save the DVM values.

8. Click the **Publish** button to activate all the changes in the editing session.

   Enter an optional note for the session in the **Publish Session** dialog.

   Changes take effect immediately upon publishing the session.

   Click **Discard** to discard your changes.

   Click **Exit** to exit your current session. Note that your session will still be open if you exit. Use **Discard** to end your session without making changes.

Configuring Silver Spring Networks Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Silver Spring Networks.

This section outlines some of the extendable lookups that must be configured for use with the Silver Spring Networks adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system. Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name**: The event name used by the head-end system.
- **Description**: A description of the device event.
- **Status**: The status of the lookup value (can be Active or Inactive).
- **Standard Event Name**: The standard event name for device events of this type, from the “Standard Event Name” extendable lookup.

UOM Code to Standard UOM Mapping

Usage received from a utility’s head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The UOM Code to Standard UOM Mapping extendable lookup is used for this purpose. Each value defined for the UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-End Unit of Measure**: The unit of measure code used by the head-end system.
- **Description**: A description of the unit of measure code.
- **Status**: The status of the lookup value (can be Active or Inactive).
- **Unit of Measure**: The unit of measure defined in the system.

Interval Status Code to Condition Mapping

Interval usage received from the Silver Spring Networks UtilityIQ can include Silver Spring Networks interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The Silver Spring Networks Interval Status Code to Condition Mapping extendable lookup is used to determine how to map Silver Spring Networks interval status codes to standard status codes when receiving usage from the Silver Spring Networks UtilityIQ.

Each value defined for the Silver Spring Networks Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status**: The Silver Spring Networks interval status code
- **Description**: A description of the interval status code.
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Condition**: The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.

Other Extendable Lookups

Business Object Name	Description
D7-ActivationStatusLookup	SSN - Activation Status
D7-ChannelStatusLookup	SSN - Channel Status
D7-DataTimeZoneLookup	SSN - Data Time Zone
D7-DeviceStatusLookup	SSN - Device Status
D7-ExecutionStatusLookup	SSN - Execution Status
D7-HeadendSQILookup	SSN - Head-End SQI to Standard SQI
D7-HeadendTOULookup	SSN - Head-End TOU to Standard TOU
D7-IntervalStatusLookup	SSN - Interval Status
D7-MeterConnectionStatusLookup	SSN - Meter Connection Status
D7-MeterReadTypeLookup	SSN - Meter Read Type
D7-OldAdminStateNameLookup	SSN - Old Admin State Name
D7-PriorityLookup	SSN - Priority Status
D7-ProvisioningActionLookup	SSN - Provisioning Action
D7-ProvisioningCommandStatus	SSN - Provisioning Command Status
D7-ProvisioningDataType	SSN - Provisioning Data Type
D7-ReadModeLookup	SSN - Read Mode
D7-RegisterReadSourceLookup	SSN - Register Read Source
D7-RegisterStatusLookup	SSN - Register Status
D7-SSNVersionLookup	SSN - Version
D7-StatusFlagNameLookup	SSN - Status Flag Name

Using the Silver Spring Networks Test Harness

Oracle Utilities Smart Grid Gateway Adapter for Silver Spring Networks (SSN) includes a test harness that can be configured to simulate the Silver Spring Networks UtilityIQ head-end system for testing the two-way commands. The test harness includes a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. See Using Smart Grid Gateway Test Harnesses for more information.

Using Smart Grid Gateway Test Harnesses

The Oracle Utilities Smart Grid Gateway Adapter includes test harnesses for each supported head-end system. The test harnesses can be configured to simulate a head-end system for testing one-way or two-way commands. The test harnesses are Multispeak 3.0 standard compliant and include a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. This section describes the general components of the test harnesses.

NOTE: This section applies to on-premises implementations of Oracle Utilities Smart Grid Gateway only. This section does NOT apply to Oracle Utilities cloud services. See Smart Grid Gateway Adapters and Oracle Utilities Cloud Services for more information about implementing Smart Grid Gateway Adapters with Oracle Utilities cloud services.

Test Harness Design

Each test harness is divided into two main layers: a front end and back end layer. The front end set of services implements the head-end system interfaces, which receive requests corresponding to the specifications shown in the list below. Each of these services calls into the "back end" layer of the test harness, which defines meters and sets their attributes. These meters are stored in a file within the test harness called meterdb.xml.

NOTE: The meterdb.xml file can be modified pre-deployment, but not post-deployment. However, the test harness retains an in-memory "database" of the meters in the file. The in-memory representation can be modified using the utility web services. Note that any changes to the in- memory structure will be lost when the server is restarted or the test harness composite is redeployed.

Itron OpenWay

- www.itron.com.ami.2008.10.control
  - ReconnectMeter
  - GetReconnectMeterResult
  - DisconnectMeter
  - GetDisconnectMeterResult
- www.itron.com.ami.2008.10.data
  - ContingencyReadByEndpoint
  - GetContingencyReadByEndpointResult
  - InterrogateByGroup
  - GetInterrogateByGroupResult
- www.itron.com.ami.2008.10.control
  - PingByEndpoints
  - ReconnectMeter
  - GetReconnectMeterResult
  - DisconnectMeter
  - GetDisconnectMeterResult
- www.itron.com.ami.2012.03.control.diagnostic
  - PingByEndpoints
  - GetPingByEndpointsResult
- www.itron.com.ami.2009.08.provisioning
  - AddMeterDefinitions
  - DeregisterMeters

Landis+Gyr

- MR_CB (Meter Reading_Customer Billing)
  - MeterAddNotification

    - MeterRemoveNotification
    - InitiateMeterReadByMeterNumber
- CD_CB (Connect/Disconnect_Customer Billing)
- InitiateConnectDisconnect

The LG Harness will send below responses to corresponding BPEL composites:

- CB_MR (Customer Billing_Meter Reading)
- ReadingChangedNotification
- CB_CD (Customer Billing_Connect/Disconnect)
- CDStateChangedNotification

Sensus RNI

- http://www.multispeak.org/Version_4.1_Release MR_Server
    - MeterAddNotification
    - MeterRemoveNotification
    - InitiateMeterReadingsByMeterID
- http://www.multispeak.org/Version_4.1_Release CD_Server
    - IitiateConnectDisconnect
- http://www.multispeak.org/Version_4.1_Release CB_Server
    - CDStatesChangedNotification (async reply)
    - ReadingChangedNotification (async reply)
- http://www.multispeak.org/Version_4.1_Release OD_Server
    - InitiateOutageDetectionEventRequest

Silver Spring Networks

- urn:com:ssn:schema:service:v1.4:DataAggregation and urn:com:ssn:schema:service:v1.6:DataAggregation
    - getMeterFieldStatus
- urn:com:ssn:schema:service:v1.4:DeviceManager and urn:com:ssn:schema:service:v1.6:DeviceManager
    - FindDevice
    - ReplaceDeviceAtLocation
    - ReplaceLocation
- urn:com:ssn:schema:service:v1.4:DeviceResults and urn:com:ssn:schema:service:v1.6:DeviceResults
    - getRemoteProvisioningResultsByJobID
    - getMeterReadResultsByJobID
    - getPingResultsByJobID
- urn:com:ssn:schema:service:v1.4:JobManager and urn:com:ssn:schema:service:v1.6:JobManager
    - addRemoteProvisioningJob
    - getJobStatus
    - addMeterReadJob
    - addPingJobfindJob

- getJobStatusForDevice

Locating the WSDL for the Test Harness

Follow these procedures to locate the test harness web service definition language (WSDL):

How to Use Enterprise Manager to Locate the WSDL

1. Open Enterprise Manager and use the navigation pane to open the dashboard of the test harness composite:
2. The top bar of the dashboard contains several buttons and icons. One of these is a “world” icon with a puzzle piece over it. Click this icon to display a list of the WSDLs and endpoint URIs for the composite:
3. Click the UtilService WSDL URL link to see the WSDL in the browser, or right click and save it to your machine

Depending on your requirements, it may be necessary to download the associated schema found in the wsdl:types section. The URL can be pasted into a browser tab and downloaded in the same manner as the WSDL. The main schema has imported schemas that may also be required.

How to Use a Direct URL to locate the WSDL

The WSDL can be accessed without Enterprise Manager by understanding the paths used on the SOA server. In general, they have the following form:

```
http://{server name}:{port number}/soa-infra/services/{partition}/{Composite}/{Web Service}?WSDL
```

For example, the test harness WSDL can be found at the following locations for each SGG adapter.

Itron OpenWay:

```
http://{server name}:{port number}/soa-infra/services/Itron_Test/ItronTestHarness/
UtilService? WSDL
```

Landis+Gyr:

```
http://{server name}:{port number}/soa-infra/services/LG_Test/LGTestHarness/UtilService?WSDL
```

Sensus RNI:

```
http://{server name}:{port number}/soa-infra/services/Sensus_Test/Sensus/UtilService?WSDL
```

Silver Spring Networks:

```
http://{server name}:{port number}/soa-infra/services/SSN_Test/SSNTestHarness/UtilService?WSDL
```

Web Services

This section describes the web services included in the test harness BPEL composite for each supported head-end system.

General Services

This section describes the general services of the test harness composite.

LoadMeterIndex

This web service loads the data store from the internal file. By default, if the store is already in memory, it will NOT reload. This behavior can be overridden with the forceReload parameter.

Input — LoadMeterIndexInput

Part: payload

Element: LoadMeterIndexRequest

Parameter	Description
forceReload	A switch telling the system whether to reload the meter index from the configuration file. Default is false.

Output — LoadMeterIndexOutput

Part: payload

Element: LoadMeterIndexResult

Parameter	Description
loaded	A boolean value for whether or not the index was reloaded from the configuration file

Fault — UtilityFault (see UtilityFault for more details).

ViewAuditTrail

This web service returns the audit log for the entire session.

Input — ViewAuditTrailInput

Part: payload

Element: ViewAuditTrailRequest

Output — ViewAuditTrailOutput

Part: payload

Element: ViewAuditTrailResult

An Entry consisting of a timestamp and an Operation. Each entry may have an associated meter object showing the latest update.

Fault — See UtilityFault, above.

UtilityFault

Fault with similar mapping to SGG/OUAF faults:

Typically, the faultCode, faultString, faultActor, and detail/text elements will be populated.

Locate Meter Services

This section describes the locate meter web services of the test harness composite.

FindMeters

This web service queries the data store for one or more meters. The difference between GetMeter and FindMeters is GetMeter can return at most one meter and it must match the provided ID exactly. GetMeter will throw an error if the ID is not found. FindMeters can return more than one meter (when using the regex) and will not throw an error when the ID does not match any of the meters in the index.

Input — FindMetersInput

Part: payload

Element: FindMetersRequest

Parameter	Description
id	The meter ID for which to search
isRegex	The provided id can be a regex value when this parameter is true. Hint: to search for all meters in the system, use ".*" for the ID.

Output — FindMetersOutput

Part: payload

Element: FindMetersResult

Zero or more meter objects can be returned from the search

Fault — See UtilityFault. Unlike other methods, FindMeters does not throw an exception if the meter is not found. As such, it can be used to test for the existence of a Meter prior to querying for it.

IsMeterDefined

This web service queries whether a particular meter is defined in the data store.

Input — IsMeterDefinedInput

Part: payload

Element: IsMeterDefinedRequest

Parameter	Description
id	The meter ID for which to search

Output — IsMeterDefinedOutput

Part: payload

Element: IsMeterDefinedResult

Whether or not the provided ID is part of the index.

Fault — See UtilityFault. Thrown when meter id is not found.

GetMeter

This web service returns all the attributes of a single meter from the in-memory data store. The difference between GetMeter and FindMeters is GetMeter can return at most one meter and it must match the provided ID exactly. GetMeter will throw an error if the ID is not found. FindMeters can return more than one meter (when using the regex) and will not throw an error when the ID does not match any of the meters in the index.

Input — GetMeterInput

Part: payload

Element: GetMeterRequest

Parameter	Description
id	The meter ID for which to search

Output — GetMeterOutput

Part: payload

Element: GetMeterResult

The meter object requested by the ID.

Fault — See UtilityFault. Thrown when meter id is not found.

Meter Administration Services

This section describes the meter administration services of the test harness composite.

AddMeters

This web service adds a set of meters to the in-memory data store. This will not permanently add it to the control file.

Input — AddMetersInput

Part: payload

Element: AddMetersRequest

Parameter	Description
id	The identification code for the meter.
macID	A MAC address that must be unique within the system.
utility	An informational string.
serviceType	One of the valid ServiceType values (see schema). "Electric" is the only option at this time.
isCommissioned	Whether or not the meter is in a commissioned state.
loadActionCode	One of the possible LoadActionCode values used in Connect and Disconnect (see schema).
outageEventType	One of the possible OutageEventType values used in Device Status Check (see schema).
executionStatus	One of the possible ExecutionStates (see schema). These values control how the meter will respond to commands.
groupName	The name linking multiple meters together into a set.
jobExecutionStatus	One of the possible Job Execution Status values (see schema). This attribute determines how requested jobs perform.
updateIfExisting	Whether or not to update the meter with the provided values if it already exists in the index.
Comment	An informational string describing the purpose of the meter.
Channels	A listing of unit of measures supported by this meter.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.

Output — AddMetersOutput

Part: payload

Element: AddMetersResult

Whether or not each meter was added to the index.

Fault — See UtilityFault

RemoveMeter

This web service removes a meter from the in-memory data store. This will not permanently remove it from the control file.

Input — RemoveMeterInput

Part: payload

Element: RemoveMeterRequest

Parameter	Description
id	The ID for the meter to be removed

Output — RemoveMeterOutput

Part: payload

Element: RemoveMeterResult

Whether or not the meter was removed from the index.

Fault — See UtilityFault. Thrown when meter id is not found.

AddMeterChannel

This web service adds a new channel to a single meter.

Input — AddMeterChannelInput

Part: payload

Element: AddMeterChannelRequest

Parameter	Description
id	The identification code for the meter.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.

Output — AddMeterChannelOutput

Part: payload

Element: AddMeterChannelResult

Whether or not the channel was added to the index.

Fault — See UtilityFault. Thrown when meter id is not found.

RemoveMeterChannel

This web service removes a Channel from a meter.

Input — RemoveMeterChannelInput

Part: payload

Element: RemoveMeterChannelRequest

Parameter	Description
id	The ID for the meter to be removed.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.

These three parameters are combined to locate a unique channel

Output — RemoveMeterChannelOutput

Part: payload

Element: RemoveMeterChannelResult

Whether or not the channel was removed from the meter.

Fault — See UtilityFault. Thrown when meter id is not found.

ReadScalarMeter

This web service generates a scalar reading for each channel of a given meter.

Input — ReadScalarMeterInput

Part: payload

Element: ReadScalarMeterRequest

Parameter	Description
id	The ID for the meter to be read

Output — ReadScalarMeterOutput

Part: payload

Element: ReadScalarMeterResult

Zero or more scalar readings for the given meter.

Parameter	Description
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.
value	A random number representing the scalar reading.

Meter Attribute Administration Services

This section describes the meter administration services of the test harness composite.

GetLoadActionCode

This web service queries whether the given meter is connected or disconnected. This method is used by the Connect/Disconnect service. The values for load action code are:

- Connect

- Disconnect

Input — GetLoadActionCodeInput

Part: payload

Element: GetLoadActionCodeRequest

Parameter	Description
id	The ID for the meter for which the load action code status should be retrieved

Output — GetLoadActionCodeOutput

Part: payload

Element: GetLoadActionCodeResult

The connection status of the requested meter.

Fault — See UtilityFault. Thrown when meter id is not found.

SetLoadActionCode

This web service updates the load action code for a given meter. This method is used by the Connect/Disconnect service. The values for load action code are:

- Connect
- Disconnect

Input — SetLoadActionCodeInput

Part: payload

Element: SetLoadActionCodeRequest

Parameter	Description
id	The ID for the meter for which the load action code status should be set.
value	The new value of LoadActionCode to set on the meter.

Output — SetLoadActionCodeOutput

Part: payload

Element: SetLoadActionCodeResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault — See UtilityFault. Thrown when meter id is not found.

IsCommissioned

This web service queries the commissioning status for a given meter. This service is used by the Commission/ Decommission process. The commissioning attribute can be true or false.

Input — IsCommissionedInput

Part: payload

Element: IsCommissionedRequest

Parameter	Description
id	The ID for the meter for which the Commissioned status should be retrieved

Output — IsCommissionedOutput

Part: payload

Element: IsCommissionedResult

The value of the Commissioned status attribute for the requested meter.

Fault — See UtilityFault. Thrown when meter id is not found.

SetCommission

This web service updates the commissioning status for a given meter. This service is used by the Commission/ Decommission process. The commissioning attribute can be true or false.

Input — SetCommissionedInput

Part: payload

Element: SetCommissionedRequest

Parameter	Description
id	The ID for the meter for which the Commissioned status should be set
value	The new value of Commissioned status to set on the meter

Output — SetCommissionedOutput

Part: payload

Element: SetCommissionedResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault — See UtilityFault. Thrown when meter id is not found.

GetExecutionStatus

This web service queries the status of the property controlling the overall execution of the command. The possible values of execution status are:

- Success - The command should complete successfully
- ResponseTimeout - The asynchronous response will never arrive
- SyncOperationFail - A simulated fault will occur in the during the initial request
- AsyncOperationFailure - A simulated fault will occur in the asynchronous response

Input — GetExecutionStatusInput

Part: payload

Element: GetExecutionStatusRequest

Parameter	Description
id	The ID for the meter for which the ExecutionStatus should be retrieved

Output — GetExecutionStatusOutput

Part: payload

Element: GetExecutionStatusResult

The value of the ExecutionStatus attribute for the requested meter.

Fault — See UtilityFault. Thrown when meter id is not found.

SetExecutionStatus

This web service updates the property controlling the overall completion of the command. The possible values of execution status are:

- Success - The command should complete successfully
- ResponseTimeout - The asynchronous response will never arrive
- SyncOperationFail - A simulated fault will occur in the during the initial request
- AsyncOperationFailure - A simulated fault will occur in the asynchronous response

Input — SetExecutionStatusInput

Part: payload

Element: SetExecutionStatusRequest

Parameter	Description
id	The ID for the meter for which the ExecutionStatus should be set
value	The new value of ExecutionStatus to set on the meter

Output — SetExecutionStatusOutput

Part: payload

Element: SetExecutionStatusResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault — See UtilityFault. Thrown when meter id is not found.

Creating a Custom Adapter for Smart Grid Gateway

Customers can create their own customized adapter for using Smart Grid Gateway with a specific head-end system. This can be done using the Oracle Utilities Smart Grid Gateway Adapter Development Kit. This section describes the Adapter Development Kit and its components.

> **NOTE:** Unless otherwise noted, the contents of this section apply to on-premises implementations of Oracle Utilities Smart Grid Gateway only, and do NOT apply to Oracle Utilities cloud services. See Smart Grid Gateway Adapters and Oracle Utilities Cloud Services for more information about implementing Smart Grid Gateway Adapters with Oracle Utilities cloud services.

Adapter Development Kit Overview

The Oracle Utilities Smart Grid Gateway (SGG) Adapter Development Kit provides a starting point for customers to create their own customized adapter for using SGG with a specific head-end system. SGG uses Oracle Service Bus (OSB) and Oracle Business Process Execution Language (BPEL) to facilitate communication with other head-end systems. The following functionality can be configured:

Measurement Data and Device Event Loading: Data parsing and transformation via Oracle Service Bus from the smart meter format into the Oracle Utilities Service and Measurement Data Foundation unified format for measurement data and device events.

Measurement Data and Device Event Processing: Configurable mapping for meter status codes and device event names to Oracle Utilities Service and Measurement Data Foundation standard values.

Commissioning Communication: Business objects and BPEL processes to support the Meter Add Notification message.

Connect Communication: Business objects and BPEL processes to support Initiate Connect/Disconnect and Connect/Disconnect State Change Notification messages.

Disconnect Communication: Business objects and BPEL processes to support Initiate Connect/Disconnect and Connect/Disconnect State Change Notification messages.

Decommissioning Communication: Business objects and BPEL processes to support Meter Removal Notification message.

On-Demand Read: Business objects and BPEL processes to support Initiate Meter Read by Meter Number and Reading Changed Notification messages.

Device Status Check: Business objects and Oracle BPEL processes to support the outbound Initiate Outage Detection communication, the inbound Outage Detection Event Notification communication, and the processing of Event Notifications.

What Does the Adapter Development Kit Provide?

The Oracle Utilities Smart Grid Gateway Adapter Development Kit includes the following:

Sample Oracle Service Bus Processes: A file parsing and transformation process framework, delivered as an Oracle Service Bus (OSB) configuration, that can be used as a reference for creating the functionality to import usage readings and device events from a specific head-end system.

Sample Oracle Business Process Execution Language Processes: A set of sample communication processes that support Remote Connect, Remote Disconnect, Commission, Decommission, Device Status Check, and On-Demand Read commands delivered as an Oracle Business Process Execution Language (BPEL) project.

Oracle Utilities Smart Grid Gateway Business Objects: Business objects to support measurement data and device event loading and two-way commands such Commission, Decommission, Remote Connect, Remote Disconnect, and Device Status Check.

Sample Two-Way Communication Test Harness: A head-end system emulator delivered as a BPEL composite.

Demonstration Data: A set of sample data provided in the demonstration database that can be used with the adapter processes as delivered (prior to any customization you might perform as part of your implementation).

Adapter Development Kit Processing

This section provides details concerning the OSB processing, BPEL Processes, and OUAF objects supplied as part of the base package. This information illustrates how the base package objects were designed, and can serve as the basis for any customizations you create as part of your implementation.

Initial Measurement Data and Device Event Loading

The initial measurement data load and subsequent device event processing use OSB to poll for, parse, and transform the head-payloads into the Oracle Utilities Smart Grid Gateway service format. Payloads contain measurements and meter events in some head-end specific format OSB then places each service call into a JMS queue within the Oracle Utilities applications. The JMS client consumes the entries and invokes the respective services in parallel then a service creates initial measurements with data in a common format with head-end-specific processing as needed. A second service creates device events with data in a common format.

Initial Measurements and Device Events

The usage and event data exported from the AMI head-end system is loaded into Oracle Utilities as initial measurement and device event data. You can customize processing of this by configuring the following base product OSB projects:

1. **SGG-DG-CSV-BASE** contains components that are not to be changed on customer site. They implement functions specific to the CSV format processing such as validation and transformation.
2. **SGG-DG-CSV-CM** allows for customization and simplifies future upgrades.
3. **SGG-DG-XML-BASE** contains components that are not to be changed on customer site. They implement functions specific to the IMD and event upload format processing such as validation and transformation.
4. **SGG-DG-XML-CM** allows for customization and simplifies future upgrades.

The runtime configuration settings for the SGG-DG-CSV-CM project are stored in the EnvironmentSettings.xq xquery file. You can use this file to adjust initial measurement and device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the SGG-DG-CSV-CM EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRaw	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementEvent	Holds the name of inbound web service for the device event seeder.	
destinationRootElementIMD	Holds the name of inbound web service for the IMD seeder.	

The runtime configuration settings for the SGG-DG-XML-CM project are stored in the EnvironmentSettings.xq xquery file. You can use this file to adjust initial measurement and device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the SGG-DG-XML-CM EnvironmentSettings.xq file:

Element	Description	Valid Values
populateRawIMD	Determines if the initial measurement data is populated as raw data.	true false
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElement	Holds the name of inbound web service for the IMD seeder.	

For additional information about the sample OSB implementation included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit, see Oracle Service Bus Processing.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service for the SGG-DG-CSV-CM and SGG-DG-XML-CM projects. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

Processing Data in the Adapter Development Kit Native Format

Usage and event data exported from the AMI head-end system can be loaded into Oracle Utilities in the “native” initial measurement and device event data formats (the format of the initial measurement and device event seeder business objects). You can customize processing of this by configuring the following base product OSB projects:

1. **SGG-DG-SEEDER-BASE** contains components that are not to be changed on customer site. They implement functions specific to the IMD and event upload format processing such as validation and transformation.
2. **SGG-DG-SEEDER-CM** allows for customization and simplifies future upgrades.

The runtime configuration settings for the SGG-DG-SEEDER-CM project are stored in the EnvironmentSettings.xq xquery file. You can use this file to adjust initial measurement and device event data processing. For example, if you want to load raw data you would specify “true” for the content of the populateRaw element.

The following table describes the elements included in the SGG-DG-SEEDER-CM EnvironmentSettings.xq file:

Element	Description	Valid Values
callPreProcessing	Determines if the preprocessing proxy service is called.	true false
callPostProcessing	Determines if the postprocessing proxy service is called.	true false
destinationRootElementInterval	Holds the name of inbound web service for the interval IMD seeder.	
destinationRootElementScalar	Holds the name of inbound web service for the scalar IMD seeder. In most cases it is the same as destinationRootElementInterval.	
destinationRootElementEvent	Holds the name of inbound web service for the device event seeder.	
publishServices/service	Specifies the name of the business service within the OSB project used to publish data for external systems (such as Oracle DataRaker).	
filterEvents	Determines if events should be filtered.	true false
filterUsage	Determines if usage should be filtered.	true false

See The Adapter Development Kit Native Format for more information about the ADK “native” format.

Publishing Initial Measurement Data and Device Events

The Adapter Development Kit can be configured to publish initial measurement data and device events for use in Oracle DataRaker or other external systems. This functionality is supported through a combination of OSB components and BPEL composites.

Enabling Data Publishing

Publishing data is enabled by referencing a publisher business service in the publishServices/service element in the EnvironmentSettings.xq file as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

The following components provided with the SGG-DG-SEEDER-CM OSB project are used in publishing measurement data and device events to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This is the business service that should be specified in the EnvironmentSettings.xq file.
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Configuring Publishing Output

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems.

Initial measurement and device event data is published in the "native" initial measurement data format (the format of the initial measurement and device event seeder business objects). This format includes normalized unit of measure, condition, and device event codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published via this feature is published prior to VEE processing. In addition, filtering can NOT be applied to initial measurement or device event data published via this feature.

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Filtering Initial Measurement Data

The Adapter Development Kit can be configured to filter initial measurement data passed into Oracle Utilities Smart Grid Gateway and Meter Data Management. Filtering data is enabled by setting the <filterUsage> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterUsage>true</filterUsage>
```

When filtering is enabled, only measurements whose <externalUOM> matches one of the values defined in the **ADK - UOM Code to Standard UOM Mapping** extendable lookup (DG-HeadendUOMLookup) are passed into the system for processing.

Filtering Events

The Adapter Development Kit can be configured to filter device events passed into Oracle Utilities Smart Grid Gateway and Meter Data Management. Filtering data is enabled by setting the <filterEvents> element in the EnvironmentSettings.xq file to “true” as follows:

```
<filterEvents>true</filterEvents>
```

When filtering is enabled, only device events whose <externalEventName> matches one of the values defined in the **ADK - Device Event Mapping** extendable lookup (DG-DeviceEventMappingLookup) are passed into the system for processing.

Prioritized Device Event Processing

The SGG adapter prioritizes processing of device events created from smart meter commands and/or completion events by setting the Execution Method flag in these types of device events to “Real Time” (D1RT). In addition, device events received with the Execution Method flag set to “Real Time” will be processed in real time rather than via batch processing. See **Device Event Prioritization** in the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide* for more information.

Configuring Payload Processing Notifications

By default, Payload Statistics, Payload Error Notification, and Payload Summary notification activities are sent to the Notification Queue within Oracle Service Bus. Smart Grid Gateway adapters can be configured to restrict sending these activities by defining optional JCA transport dynamic properties in the appropriate OSB project. The following optional JCA transport dynamic properties can used to control if these notification activities are sent:

- **SGG_SEND_STATISTICS**: Controls sending Payload Statistic activities (true: send, false: do not send)
- **SGG_SEND_ERROR**: Controls sending Payload Error Notification activities (true: send, false: do not send)
- **SGG_SEND_SUMMARY**: Controls sending Payload Summary activities (true: send, false: do not send)

When these properties are set to false, the corresponding payload notification will not be sent to the JMS Queue. Note these properties are not populated by default, and default to true if omitted.

To create these properties:

1. Navigate to the SGG-D3-CIM-EVENT-CM project in the Oracle Service Bus Console.
2. Expand the **Proxy Services** folder.
3. Select the **InboundProxyService** proxy service (not the Pipeline or WSDL).
4. Click **Create** to create a new session.
5. Select **Transport Details**.
6. Click the plus sign in the **Dyanmic EndPoint Properties** section. An empty row will appear.
7. Enter the property you wish to define in the **Property** column.
8. Enter "false" in the **value** colulmn.

9. Repeat steps 6-8 for each property you wish to define.

10. Click **Activate** to save and activate your changes.

NOTE: These properties are only used if the SGG_EXT_ID_PATTERN is not present or the regular expression captures nothing. The SGG_EXT_ID_PATTERN will override these properties when the regular expression captures group data.

Base Package Business Objects

The adapter development kit base package includes the following device and initial measurement business objects:

Business Object Name	Description
DG-InitialLoadIMDInterval	Generic Initial Load IMD - Interval Used when loading customized adapter interval measurements into the system for the first time.
DG-InitialLoadIMDScalar	Generic Initial Load IMD - Scalar
DG-SmartMeter	ADK — Smart Meter

Device Communication

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, device status check, or an on-demand read. The designated BPEL process transforms the request from Oracle Utilities format to MultiSpeak format and invokes the related head-end system web service. The head-end system then returns a reply, and the BPEL process transforms the reply message back to the appropriate format so that Oracle Utilities can receive the response.

Communication Flows

The table below lists the communications created for each Adapter Development Kit command:

Command	Outbound Communication	Inbound Communication	Completion Event
Remote Connect	ADK - Initiate Connect or Disconnect	ADK - Connect/ Disconnect State Change	Connect Device Create IMD
Remote Disconnect	ADK - Initiate Connect Disconnect	ADK - Connect/ Disconnect State Change	Disconnect Device Create IMD
Device Commissioning	ADK - Meter Add Notification		Device Commissioning
Device Decommissioning	ADK - Meter Remove Notification		Device Decommissioning
On-Demand Read (Scalar)	ADK - Initiate Meter Read by Meter ID	ADK - Reading Changed Notification	Create IMD
On-Demand Read (Interval)	ADK - Initiate Meter Read by Meter ID	ADK - Reading Changed Notification	Create IMD
Device Status Check	ADK - Initiate Outage Detection	ADK - Outage Detection Event Notif	

Device Communication Base Package Business Objects

The adapter development kit base package includes the following communication business objects:

Business Object Name	Description
DG-ConnectDisconStateChgNtf	Generic - Connect/Disconnect State Change
DG-InitiateConnectDisconnect	Generic - Initiate Connect Disconnect
DG-InitiateMeterByMeterId	Generic - Initiate Meter Read By Meter ID
DG-InitiateOutageDetection	Generic - Initiate Outage Detection
DG-MeterAddNotification	Generic - Meter Add Notification
DG-MeterRemoveNotifcation	Generic - Meter Remove Notification
DG-OutageDetectEvtNotification	Generic - Outage Detection Event Notification
DG-ReadingChgNotification	Generic - Reading Changed Notification

Event Data Mapping

The head-end system event file format can map as follows into the business object, D1-DeviceEventMappingLookup:

Head-End System Flat File Field	Device Event Seeder BO Element	Comments
Transaction ID (from Header record)	External Source Identifier	This is the file name.
Device Identifier	External Device Identifier	
Event Name	External Event Name	
Event Creation Date/Time	Event Date/Time	
Device Type	External Device Type	This element has no real bearing on the device type within MDM/SGG. Its valid values include (although the element itself is free-form): Meter Collector Router
Service Location ID	External Service Location ID	
Communication Module Serial Number	External Communication Module Identifier	
Event Category ID	External Event Category	
Event Severity	External Event Severity	Valid values include (although the element itself is free-form): Alert Information
Status Value	External Status Value	This represents additional information that relates to the event itself.
Status Date/Time	External Status Date/Time	The date & time at which the additional information referenced above had occurred.

External System

You must create an External System for each external system to which Oracle Utilities Smart Grid Gateway will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch or Real-time)

- The corresponding message senders
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

Inbound / Outbound Service Configuration

The inbound/outbound message utility allows you to configure your system to receive information from and to send information to external applications using XML. The adapter for Smart Grid Gateway uses one inbound web service to map device events. This is the same inbound web service used by the D1 application.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send.

The Oracle Utilities Smart Grid Gateway Adapter Development Kit includes the following inbound web services:

Inbound Web Service	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Used for upload of device events. The Device Event Seeder business object serves as a means of adding device events both from outside the application and from online. Its pre-processing algorithms determine the device event type - which in turn defines the device event BO that should be used to create the device event. If a device event type can't be determined, the device event is created using this BO. Such a device event can then be re-processed - and if successful, a new device event is created.
D1-InitialLoadIMD	Used for initial measurement upload. The IMDSeeder business object is used to determine the type of initial measurement business object to instantiate when receiving usage readings from a head-end system.
D1-DeviceStatusCheck	Device Status Check This service is invoked by the integration layer to instantiate a Device Status Check command.
D1-InitialLoadIMD	Used by OSB to instantiate an IMD This inbound web service is used by OSB to instantiate an Initial Measurement Data for incoming interval usage in the Generic format.
D1-RemoteConnect	Remote Connect This service is invoked by the integration layer to instantiate a Remote Connect command.
D1-RemoteDisconnect	Remote Disconnect This service is invoked by the integration layer to instantiate a Remote Disconnect command.

Inbound Web Service	Description
DG-ConDisconStChgNotification	Initiate Connect Disconnect response. Retrieves asynchronous response from Initiate Connect Disconnect command.
DG-OutageDetectionEventNotification	Initiate Outage Detection Response Retrieve response from the Initiate Outage Detection Event Notification command.
DG-ReadingChangedNotification	Reading Changed Notification Notification that a device reading has changed.

Message Senders

Message senders define the details of how messages are sent to an external system. As in the case of outbound communication business objects and outbound message types, the set of message senders you need to create is based on the types of messages the system is designed to accept.

The Oracle Utilities Smart Grid Gateway Adapter Development Kit includes the following message senders:

Message Sender	Description
DG-COMM	Generic Adapter Commission
DG-DCOMM	Generic Adapter Decommission
DG-INTOUDET	Generic Adapter — Initiate Outage Detection Request
DG-XAISender	Commission / Decommission Sender

BPEL Processes

These processes are responsible for performing the conversion from Oracle Utilities format to MultiSpeak 4.1 format, invoking process callouts and invoking the remote endpoint to trigger the device events.

OnDemandRead Composite Process: Invokes the remote endpoint to trigger the on-demand read event. An asynchronous reply responds to the OUAF layer when the reading arrives.

ConnectDisconnect Composite Process: Invokes the remote endpoint to trigger the connect/disconnect event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

CommissionDecommission Composite Process: Invokes the remote endpoint to trigger the commission or decommission event. After the synchronous call completes, a one of the following second business callout services is invoked to determined if the related “received” or “completed” callout should be executed:

- isExecutingCommissionReceived-Callout
- isExecutingCommissionCompleted-Callout
- isExecutingDecommissionReceived-Callout
- isExecutingDecommissionCompleted-Callout

DeviceStatusCheck Composite: Invokes the remote endpoint to trigger the initiate outage detection event. An asynchronous reply responds to the OUAF layer when confirmation of the requested event arrives.

ProcessCallout Composite: This business callout provides a point at which customers and implementers can incorporate custom business logic and transformations. This composite includes the WSDLs and processing logic for all of the MultiSpeak processes. The default implementation of each method is a direct return of the input.

For additional information about the BPEL processes included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit, see Business Processing Execution Language Processing.

Web Services

These web services are all defined in the head-end system. The WSDLs were added to a Meta Data Storage (MDS) layer in OUAF and all references to the WSDL point to this MDS location. These web services have HTTP security by default. You may need to modify the security as a part of your implementation.

Web Service	Related BPEL Process	Description
CB_ServerService	ConnectDisconnect	This web service defines the return interface, the means by which the status is returned to the calling system. This web service is only be invoked by the head end system, not OUAF. Only the CDStateChangeNotification web method is implemented in the composite. The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic Adapter/ConnectDisconnect/CB_ServerService
CB_Server	OnDemandRead	This web service defines the return interface, the means by which the reading is returned to the calling system. This web service is only be invoked by the head end system, not OUAF. Only the ReadingChangedNotification web method is implemented in the composite. The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic Adapter/OnDemandRead/CB_Server
OA_ServerService	DeviceStatusCheck	This web service defines the asynchronous return for InitateOutageDetectionEventRequest for solicited responses. This web service is only be invoked by the head end system, not OUAF. Only the ODEventNotification, PingURL, and GetMethods web methods are implemented in the composite. The endpointURI format is: http://<EM_SERVER>:<EM_SERVER_PORT>/soa-infra/services/Generic Adapter/DeviceStatusCheck/OA_ServerService

For additional information about the web services included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit, see Business Processing Execution Language Processing.

Oracle Service Bus Processing

This chapter describes how to use Oracle Service Bus (OSB) and the components of the Oracle Utilities Smart Grid Gateway Adapter Development Kit to develop an adapter for importing usage reading and device events.

OSB Overview

The examples in this chapter provide an implementer with the information required to create a customized adapter using the Oracle Utilities Smart Grid Gateway Adapter Development Kit. Customized adapters are required in situations where a productized Smart Grid Gateway adapter is not available for a particular head-end system. The examples below explain how to receive and process inbound meter usage and events. The examples contain two parts: the incoming file parsing functionality implemented in Java, and an easy, configurable and customizable Oracle Service Bus (OSB) configuration, including an implementation of post-parsing functionality.

OSB Prerequisites

Development of an Oracle Utilities Smart Grid Gateway adapter requires a number of technical skills as well as a development environment.

Required Technical Skills

The following technical skills are required for developing an adapter using the development kit:

- Experience in Java programming
- Experience in development using XQuery and XPath
- Experience in XML programming
- Experience in OSB development and administration
- Understanding of Oracle Utilities Service and Measurement Data Foundation (SMDF) architecture

Environment Requirements

Development of the Oracle Utilities Smart Grid Gateway adapter requires an environment configured as follows:

- The Oracle WebLogic server must be installed.
- The OSB must be installed.
- An OSB domain must be created.
- Java Messaging Service (JMS) queues for Initial Measurement Data (IMD) and Notification messages must be created.
- The spl-d1-osb-2.0.1.jar containing the com.splwg.d1.sgg.osb.common Java package must be available. This is a single jar for all vendor-specific adapters.
- The archive sgg-osb-generic-adapter.zip must be available. This file is downloaded as a part of adapter development kit deployment. It contains the FileParser folder with sample source code and the com.splwg.dg.sgg.osb.configuration.jar file with sample OSB projects.

OSB Related Documentation

For more information about using the adapter development kit and Oracle Service Bus, see the following Oracle documents:

- *Oracle Fusion Middleware User's Guide for Technology Adapters*

- *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*
- *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*
- *Oracle Fusion Middleware User's Guide for Oracle JDeveloper*
- *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide*
- *Oracle Utilities Meter Data Management / Smart Grid Gateway Administrative User Guide*

OSB Processing Terms and Acronyms

This section lists several terms and acronyms used throughout this chapter.

Term	Definition
AMI	Advanced Metering Infrastructure
CSV	Comma-Separated Values
FA	File Adapter. Oracle JCA File Adapter.
FP	File Parser, Java code that parses input file and generates Plain XML
GA	Smart Grid Gateway generic adapter
GFP	Generic File Processor, Java code that is invoked by FA. It communicates with FP
IMD	Initial Measurement Data
IMD XML	The final XML containing the IMD.
JAR	Java Archive
JAXB	Java Architecture for XML Binding
JCA	Java Connector Architecture
JDeveloper	Freeware Oracle integrated development environment for development of Java-based SOA and Java EE applications
JMS	Java Message Service
MDB	Message Driven Bean
SMDF	Service and Measurement Data Foundation
OSB	Oracle Service Bus
OUAF	Oracle Utilities Application Framework
Plain XML	An intermediate XML containing all of the values from the inbound file. It is necessary because OSB Message Flow can handle only XML.
PPS	Processing Proxy Service, OSB proxy service where the validation and transformation of Plain XML is implemented. It is necessary to catch errors that occurred before Plain XML is transformed to IMD or Device Event seeder structures.
RFD	Rejected File Descriptor, File containing information about a portion of input file that was a cause of error.
RPPS	Result Processing Proxy Service, OSB proxy service where processing of transformation result is implemented. It is necessary to catch errors that occurred after Plain XML is transformed to IMD or Device Event seeder structures.
SGG	Smart Grid Gateway
Weblogic	Oracle J2EE Application Server
WSD	Web Session Directory
XML	eXtensible Markup Language
XPath	Programming language for selecting nodes from an XML document
XQuery	Programming language to query XML.

Oracle Utilities Smart Grid Gateway Adapters

This section contains detailed information on the different components that make up an Oracle Utilities Smart Grid Gateway adapter and the application logic that is needed to implement one.

Adapter Components

The following table lists the adapter components:

Component	Description
JCA File Adapter (FA)	Technology adapter for reading and writing files on the local file system. It is responsible for polling files from incoming folder and passing it to the Generic File Processor.
Generic File Processor (GFP)	Framework component that is implemented in Java. It is responsible for instantiation of the File Parser, getting Plain XML from it, and passing it to OSB message Flow. It also performs payload statistics related functionality such as gathering data and generating notification messages.
File Parser (FP)	Component implemented in Java. It is specific to every different incoming format. It is responsible for parsing the incoming file, breaking the payload into logical parts (debatching), generating Plain XML for every logical part, and returning XML back to invoking GFP. **Plain XML** could be any logical subset of data that can be mapped to IMD. It is an intermediate format/schema between the raw data and IMD XML. The Plain XML schema must be defined by the file parser developer. Raw data is read in portions from the input file and converted to Plain XML before passing to OSB message flow. The reason for data being read in portions is as follows: Assume that an input file contains readings for 100 Measuring Components. You would want to read the input file in portions for two particular reasons. 1. Reading data of exactly one measuring component ensures that it is mapped appropriately to one IMD at a time. 2. Reading data in portions ensures that the entire file is not loaded into memory, which could cause resource issues.
Inbound Proxy Service	OSB proxy service that contains FA-related configuration settings. It statically routes all messages to Processing proxy Service.
Processing Proxy Service (PPS)	OSB proxy service that validates and transforms Plain XML. This service is necessary for catching errors that occur before Plain XML is transformed to IMD or Device Event seeder structures.
Result Processing Proxy Service (RPPS)	OSB proxy service that processes transformed data. This service is necessary for catching errors that occur after Plain XML is transformed to IMD or Device Event seeder structures.

Processing Life Cycle

This section outlines the life cycle from the initial input file to the initial measurement output. Refer to OSB Processing Terms and Acronyms for descriptions of the abbreviations and acronyms used in this chapter.

See the **Logic Sequence Diagram** for general description of how the components interact.

OSB Processes

1. The JCA File Adapter starts reading the file.
2. FA instantiates and initializes the GFP.
3. FA invokes GFP passing an open stream to the incoming file.

4. GFP sends **D1-PayloadStatistics** notification message to FA.
5. GFP instantiates and initializes (if it is not done yet) the FP that is defined in the Inbound Proxy properties for parsing the file.
6. GFP invokes the FP (See File Parser Processes below).
7. In case of error in FP it invokes GFP. GFP sends **D1-PayloadErrorNotif** notification message to FA.
8. GFP gets the Plain XML as a return from FP.
9. GFP returns the Plain XML to the FA.
10. The Plain XML is passed via InboundProxyService to Processing Proxy service (PPS). In PPS's message flow, the Plain XML is validated and transformed to IMD XML.
11. PPS passes the IMD to Result Processing Proxy service (RPPS). RPPS publishes the IMD XML to a JMS queue, which is then picked up by Message Driven Bean (MDB).
12. OSB processes 3 through 9 are repeated until FP returns NULL on process 9.
13. GFP sends **D1-PayloadSummary** notification message to FA.

File Parser Processes

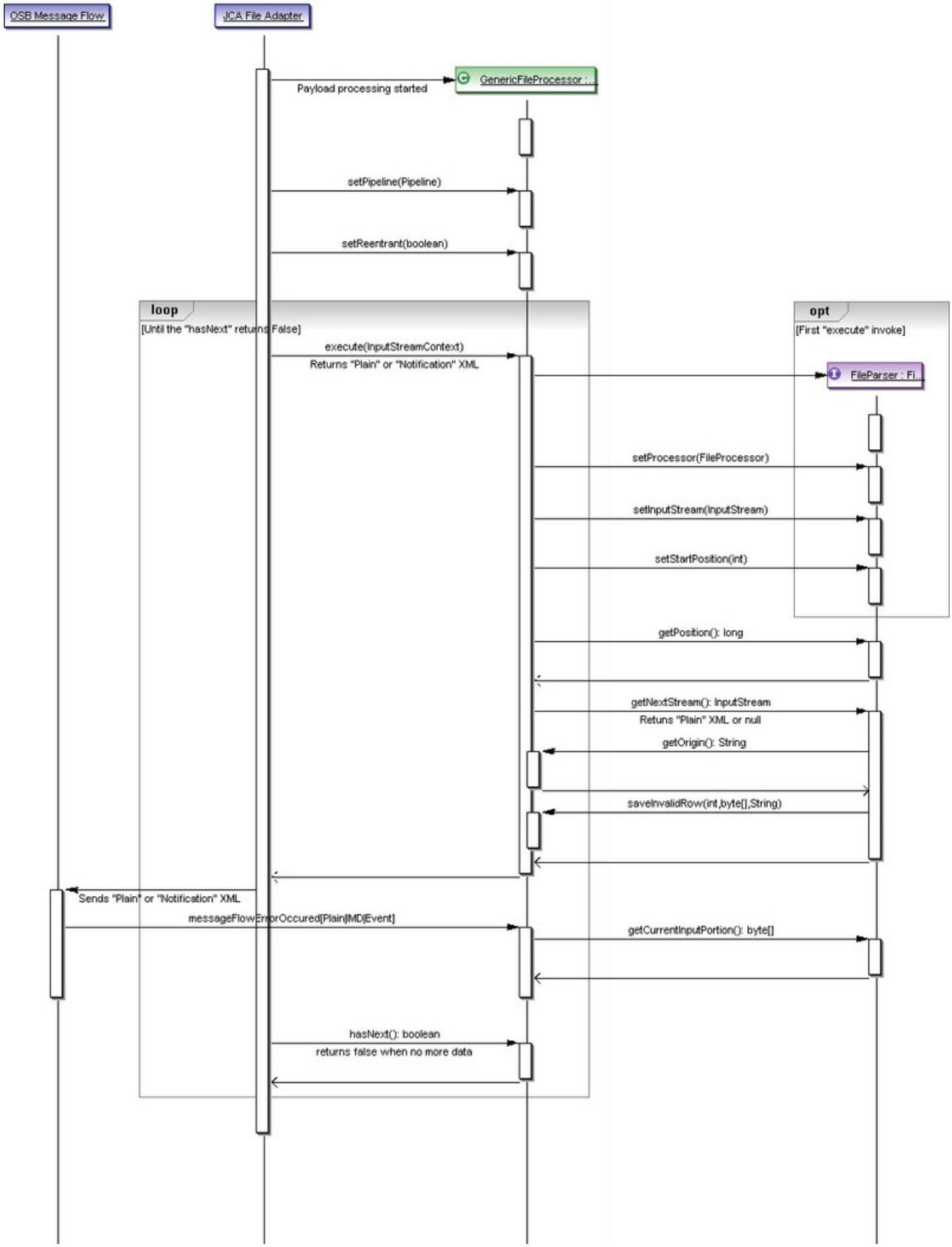
1. FP starts reading the input stream
2. FP instantiates a JAXB object of the type of Plain XML. The JAXB related interfaces and implementation classes are generated by using JDeveloper (See *Oracle Fusion Middleware User's Guide for Oracle JDeveloper* or Generating Java Classes from XML Schemas).
3. FP reads the input stream in segments until it hits a logical end of data with which it could create a Plain XML.
4. FP populates the data that is read into the Plain XML JAXB object.
5. FP marshals the JAXB object into Plain XML and passes back GFP. If the end of file (EOF) is reached FP passes back NULL.
6. File parser processes 2 through 5 are performed for every invocation by GFP.

Java Project

The source code in the example files contains an implementation of functionality to parse an incoming payload in Comma Separated Values (CSV) and XML formats. This implementation can be used as a reference for creating a project that is specific to your head-end system. The descriptions below are mostly related to CSV parsing functionality as it is a more complex and interesting example. The XML related parsing example is provided only to illustrate how XML-based input formats may also be processed.

Logic Sequence Diagram

The sequence diagram below is helpful for understanding the File Parser functionality. The custom class must implement the FileParser and FileParser2 Java interfaces described in the SGG D1 jar to be qualify as a file parser. For more information, see FileParser Interface and FileParser2 Interface.


OSB Message Flow
JCA File Adapter
Payload processing started
GenericFileProcessor :...
setPipeline(Pipeline)
setReentrant(boolean)
loop
[Until the "hasNext" returns False]
execute(InputStreamContext)
Returns "Plain" or "Notification" XML
opt
[First "execute" invoke]
FileParser : Fi...
setProcessor(FileProcessor)
setInputStream(InputStream)
setStartPosition(int)
getPosition(): long
getNextStream(): InputStream
Returns "Plain" XML or null
getOrigin(): String
saveInvalidRow(int,byte[],String)
Sends "Plain" or "Notification" XML
messageFlowErrorOccured[Plain|IMD|Event]
getCurrentInputPortion(): byte[]
hasNext(): boolean
returns false when no more data

When FA finds a file in the input directory, it instantiates and calls GFP which in turn starts an interaction with the FP. The interaction can be categorized into the following three phases:

1. Instantiation and Initialization
   - setProcessor()
   - setInputStream()
   - setStartPosition()
2. Transaction
   - getPosition()
   - getNextStream()
3. Exception Handling
   - getCurrentInputPortion()

Instantiation and Initialization Phase

The GFP is instantiated and receives an Input Stream when a file gets into the input directory. At that point an instance of the FP is instantiated. Certain methods, as specified above, are invoked on the FP to initialize.

Transaction Phase

Once the FP is initialized, GFP uses the FP instance to generate the Plain XML structure. This happens with the invocation of getNextStream() method on the FP.

On every invocation, the FP returns exactly one Plain XML. The FP invocations continues until NULL is returned from the FP. NULL indicates the end of file and an indication that parsing is complete for a specific file. GFP stops invoking getNextStream() for a particular input file when it receives a NULL from the FP.

FP uses the file input stream to read the file in portions. It could read byte-by-byte or line-by-line, but at the end of each getNextStream() method call it would have read only so much that is sufficient to create one Plain XML structure. For example, in an input file, if there is sufficient data for only one Plain XML then the first call to getNextStream() would return a Plain XML and the next one would return NULL. But if there is more than one, getNextStream() calls would continue to return Plain XML structures for every incoming portion until end of file is reached.

Exception Handling Phase

Exception handling inside the FP can be categorized into two phases: **Reactive** and **Recovery:**

Reactive Phase

This phase involves catching an exception and reporting it to the GFP. It is achieved by invoking saveInvalidRow() method on the GFP (this.fileProcessor).

Note that when an exception occurs, no further parsing is done and NULL is passed back from getNextStream().

If one fails to return NULL, errors such as “Simultaneous good and bad result returned by the File Parser” can be found in the Weblogic server log.

When an exception is reported to the GFP, it does three things:

1. Creates an XML payload D1-PayloadErrorNotif and passes it to the OSB message flow. The message is published to the Notification Queue from inside PPS of the BASE OSB project.
2. It increments the “transaction error occurred” variable by invoking the utility method inside the D1 jar. All the errors occurring during the lifetime of a file parsing are captured and are reported through the message D1-PayloadSummary - which is again published to Notification Queue.

3. Finally, it creates a data file in the error directory with the portion of raw data that is read when the error occurred. It also creates a Rejected File Descriptor (RFD) file. The raw input portion is available by utilizing getCurrentInputPortion() method, which is discussed below.

As noted in the above image, the saveInvalidRow() method takes three input parameters:

1. The position at which the parsing started. It will be written to an RFD file. This is helpful in identifying the location of an error in the input file.
2. The raw data that will be written into error data file. It should be in the same to the incoming file format and it should contain data that is read since current invocation of getNextStream() method is started. It will allow correcting troubled data and place "fixed" file into an incoming folder for further re-processing.
3. The error message that is reported in D1-PayloadErrorNotif message. Note, the content of the error message has to be defined by the file parser developer.

Recovery Phase

This phase must not be confused with recovering from a handled exception discussed above. This case is when file parsing gets completely interrupted by power failure or network interruption.

Preparation for recovery: GFP maintains an internal index to the current read position of the file. This is achieved by GFP by invoking getPosition() method on the FP before every getNextStream() method call.

```
public long getPosition() {
    long retVal = this.fileProcessor.getCurrentInputPosition();
    return retVal;
}
```

Recovery: In case of an interruption and subsequent restore, GFP sets the start position on the File Parser. For instance, if the failure had occurred at byte 415, the start position would be set to 415 so that FP starts to read from that point and not from 0. The value 415 can be referred to as the recovery point.

```
public void setStartPosition(long position) {
    this.startPosition = position;
}
```

When the file parser starts parsing again, two things must occur to ensure that it starts parsing from the recovery point:

1. Store the recovery point to the startPosition field.
2. Incorporate skip logic to start the pointer from the recovery point.

Java Implementation

The Java component of the project includes the following classes and packages:

1. **com.splwg.dg.osb.common.FileParserGenGeneric** class is a super class with members that are common for CSV and XML related functionality. It implements the com.splwg.d1.sgg.osb.common.FileParser interface (see FileParser Interface). The implementation of com.splwg.d1.sgg.osb.common.FileParser2 (see FileParser2 Interface) has been added to define the IMD & Event Upload Statistics related functionality.
2. **com.splwg.dg.osb.csv.FileParserCSV** class contains the implementation of the parsing functionality for CSV formats. It performs the following logic:
   - Parses the incoming payload.
   - Transforms incoming data to Usage or Event related structures based on incoming data type.
   - Extends those structures with file information and external service provider identifier.
   - Returns one by one that structures as a stream to caller.

3. **com.splwg.dg.osb.plain** package contains JAXB related interfaces and implementation classes corresponding to the Plain XML Schema
4. **com.splwg.dg.osb.xml.FileParserXML** class contains the implementation of parsing functionality for the "IMD and Event Online Upload" XML format (See the *Oracle Utilities Service and Measurement Data Foundation User's Guide*). It performs the following logic:
    - Parses the incoming payload.
    - Extracts information about current device.
    - Breaks the incoming data to separate initial measurement data structures.
    - Extends those structures with device information, file information and external service provider identifier.
    - Returns one by one the structures as a stream to caller.

Implementation Details

This section provides an overview of the implementation details. See the comments in the source code for more detailed information.

OSB Configuration

This section outlines the OSB projects provided with the Adapter Development Kit.

OSB Project Summary

The OSB Configuration consists of four projects. These projects can be categorized by either functionality or content:

Functionality by Incoming File Format

CSV format processing:

- SGG-DG-CSV-BASE
- SGG-DG-CSV-CM

XML format processing:

- SGG-DG-SEEDER-BASE
- SGG-DG-SEEDER-CM
- SGG-DG-XML-BASE
- SGG-DG-XML-CM

Content (by purpose of content)

Content containing business logic implementation:

- SGG-DG-CSV-BASE
- SGG-DG-SEEDER-BASE
- SGG-DG-XML-BASE

Content containing configuration settings related to the objects and variables required during the processing of payloads:

- SGG-DG-CSV-CM
- SGG-DG-SEEDER-CM
- SGG-DG-XML-CM

Project Contents

The following table describes the contents of each of these projects:

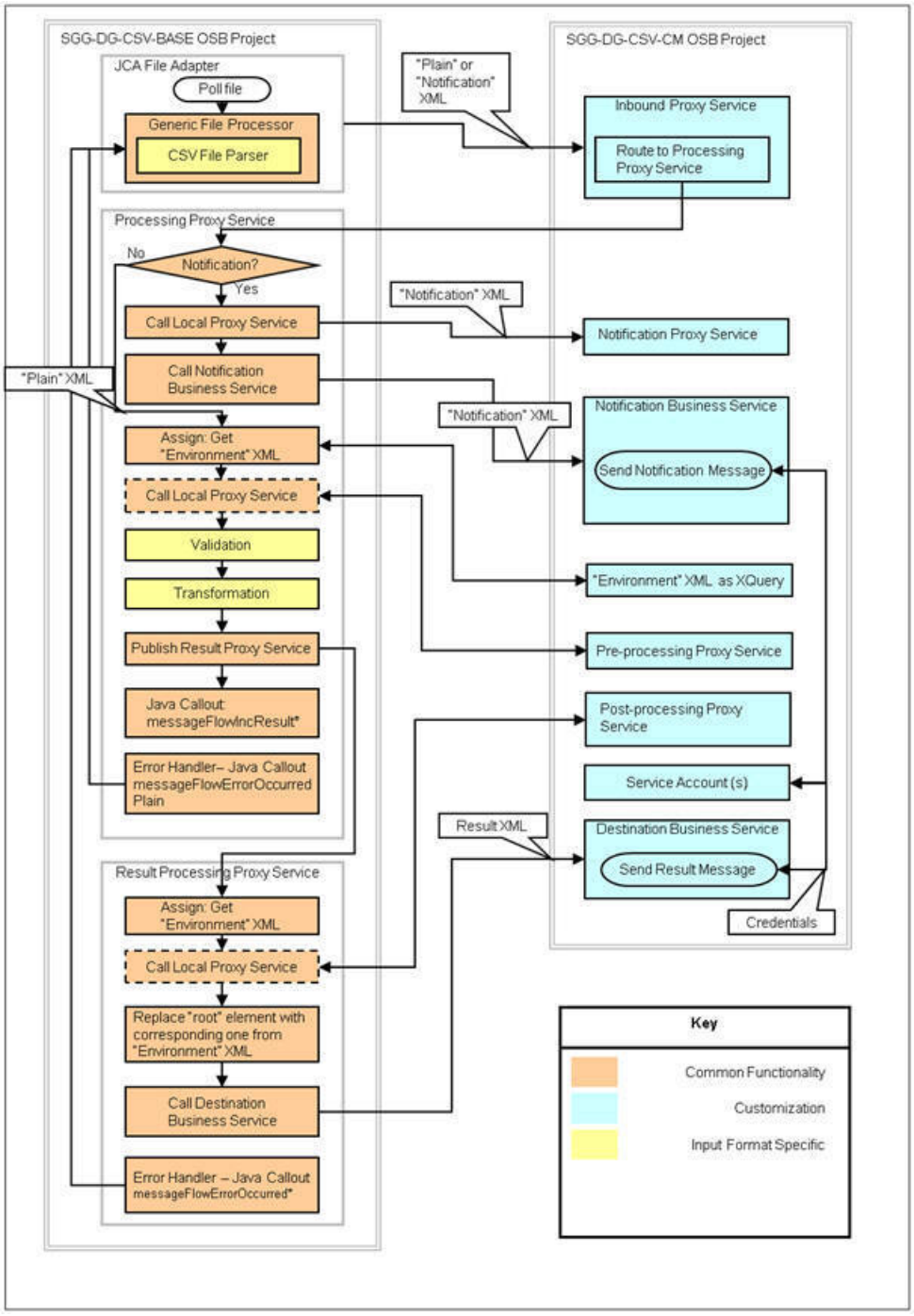
Project	Description
SGG-DG-CSV-BASE	Contains the components responsible for “actual” processing of data coming in CSV format. It can be upgraded in future releases without affecting the customization and environment settings that are stored in the SGG-DG-CSV-CM project. This project performs the following functions: • Processes notification messages that are sent from the GFP. • Calls the customizable local service proxies for pre- and post- processing of passed data • Validates passed data against the XSD schema for Plain XML format • Transforms passed Plain XML into D1-InitialLoadIMD or D1-DeviceEventSeeder formats according to type of incoming data. • Sends result structures to destination JMS queue. • Updates statistic information via calling the messageFlowIncResultIMD or messageFlowIncResultEvent method accordingly. • Handles errors that occur to provide information to the File Processing component. Error handling contains separate handlers for two stages: before and after the Plain XML is transformed to the final XML structure. The separation is necessary to make Java calls to appropriate methods (messageFlowErrorOccurredPlain, messageFlowErrorOccurredIMD or messageFlowErrorOccurredEvent) according to the current processing stage and data type.
SGG-DG-CSV-CM	Contains environment related configuration settings. It allows the customization and simplifies the future upgrades as well (See Configurng the OSB Project for example values).
SGG-DG-XML-BASE	Contains the components responsible for “actual” processing of data coming in XML format. It can be upgraded in future releases without affecting the customization and environment settings that are in SGG-DG-XML-CM project. This project performs the following functions: • Processes notification messages that are sent from GFP. • Calls the customizable local service proxies for pre- and post- processing of passed data. • Validates passed data. • Transform passed data. In current tutorial a simple XQuery code is used just to show the functionality available for transformation. • Sends result structures to destination JMS queue in form of acceptable by the D1-DeviceEventSeeder and/or D1-InitialLoadIMD inbound web services. • Updates statistic information via calling messageFlowIncResultIMD method. • Handles errors to provides information to GFP. Error handling contains separate handlers for two stages: before and after the Plain XML is transformed to the final XML structure. The separation is necessary to make a Java calls to appropriate methods (messageFlowErrorOccurred or messageFlowErrorOccurredIMD) according to the current processing stage.

Project	Description
SGG-DG-XML-CM	Contains environment related configuration settings. It allows the customization and simplifies future upgrades as well.
SGG-DG-SEEDER-BASE	Contains the components responsible for processing of data coming in “native” XML format. It can be upgraded in future releases without affecting the customization and environment settings that are in SGG-DG-SEEDER-CM project. This project performs the following functions: • Processes notification messages that are sent from GFP. • Calls the customizable local service proxies for pre- and post- processing of passed data. • Validates passed data. • Splits payload into individual initial measurements and/or device events. • Sends result structures to destination JMS queue in form of acceptable by the D1-InitialLoadIMD and/or D1-DeviceEventSeeder inbound web services. • Updates statistic information via calling messageFlowIncResultIMD method. • Handles errors to provides information to GFP. Error handling contains separate handlers for two stages: before and after the Plain XML is transformed to the final XML structure. The separation is necessary to make a Java calls to appropriate methods (messageFlowErrorOccurred or messageFlowErrorOccurredIMD) according to the current processing stage.
SGG-DG-SEEDER-CM	Contains environment related configuration settings. It allows the customization and simplifies future upgrades as well.

OSB Project Implementation Details

This section provides an overview of the implementation details. For more detailed information see the comments contained in the source code.

The following diagram shows the major processes and the data flow implemented in the OSB Configuration projects. The implementation description that follows is related to CSV parsing.



Setting Up the Adapter Environment

This section contains detailed information about the setting up of an environment with an example adapter.

Setting Up the Java Project

1. Open the sgg-osb-generic-adapter.zip file and extract the FileParser folder into a temporary location on your hard disk.
2. Create a Java project in JDeveloper and import source code into it from the folder extracted previously.

3. Add the spl-d1-osb-2.0.1.jar file to the project's Libraries and Classpath properties.
4. Deploy the project as a jar file to the destination OSB domain's lib folder.

   Note: The WebLogic server should be restarted after jar file is copied to lib folder. For more details, see *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*.

Configurng the OSB Project

1. Open the sgg-osb-generic-adapter.zip file and extract the com.splwg.dg.sgg.osb.configuration.jar file into a temporary location on your hard disk.
2. Ensure that OSB is running in the domain that is created for the adapter.
3. Import resources by using the OSB Administration Console via the **System Administration > Import/Export > Import Resources** menu item from the extracted com.splwg.dg.sgg.osb.configuration.jar file. Note that conflicts may occur during import. They will be fixed in the following steps.
4. Check and change the JCA Transport Configuration properties of the SGG-DG-CSV-CM/Proxy Services/InboundProxyService proxy service according to the deployment environment:

   Where:

   - **Endpoint Properties/SGG_INPUT_PARSER** is a name of class that implements file parsing functionality. See a Java project mentioned early.
   - **Endpoint Properties/SGG_ERROR_FOLDER** is a folder where a rejected transaction from incoming file will be placed in case when the parsing or validation is failed.
   - **Endpoint Properties/SGG_SP_EXT_REF_ID** is a value corresponding to the target Service Provider's external reference id on OUAF side. It is used in the XML structures related to the IMD & Event Upload Stats functionality. Also, this value will be placed into D1-InitialLoadIMD/serviceProviderExternalId and D1-DeviceEventSeeder/externalSenderId elements.
   - **Activation Spec Properties/IncludeFiles** is the naming convention that the Oracle File Adapter uses to poll for inbound files.
   - **Activation Spec Properties/PhisicalArchiveDirectory** is a folder where to archive successfully processed files.
   - **Activation Spec Properties/PhisicalDirectory** is an input folder to be polled.
5. Change the Endpoint URI in the Transport Configuration section of the SGG-DG-CSV-CM/Business Services/DestinationBusinessService business service according to the deployment environment with value containing URI for JMS server and queue where IMD or Device Event structures will be sent (see *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* for more detail.):
6. Change the Endpoint URI in the Transport Configuration section of the SGG-DG-CSV-CM/Business Services/NotificationBusinessService business service according to the deployment environment:
7. Update the credentials stored in the SGG-DG-CSV-CM/Service Accounts/DestinationServiceAccount service account according to the deployment environment with user ID and password required to access JMS server:
8. Repeat steps 4-7 for the SGG-DG-XML-CM project

Additional Information

This section contains additional information about the Adapter Development Kit.

Processing Large Input Files

In some environments, the OSB project may begin processing a large input file before it has been completely copied to the input directory. To prevent this, configure the MinimumAge property in the “InboundProxyService” proxy service. The MinimumAge property specifies the minimum age of files to be retrieved, based on the last modified time stamp. This enables large files to be completely copied to the input directory before they are retrieved for processing.

FileParser Interface

This section provides the source code of the FileParser interface used by the Java class delivered with the Java package described earlier in this section.

```
package com.splwg.d1.sgg.osb.common;
import java.io.IOException;
import java.io.InputStream;
/** <p>
 *  FileParser component is responsible for processing the input
 *  stream and returning a Plain XML containing all data from
 *  incoming transaction in a shape expected by the following
 *  Oracle Service Bus message flow.
 * </p>
 */
public interface FileParser {
    /**
     * Returns the next message in Plain XML format
     * or null if there are no more messages
     * @return next message as InputStream
     */
    InputStream getNextStream() throws IOException;
    /**
     * Returns the current position in the currently
     * parsed source input stream
     * @return the position in source input stream after
     * last incoming row was read and parsed
     */
    long getPosition();
    /**
     * Sets source stream
     * @param input the Input Stream that should be parsed
     */
    void setInputStream(InputStream input);
    /**
     * Sets "owner" FileProcessor
     * @param owner the Instance of FileProcessor that will
     * be notified about current processing via call to
     * utility methods (saveInvalidRow and setParseFinishNote)
     */
    void setProcessor(FileProcessor owner);
    /**
     * Sets the position from which the parsing of source
     * stream should be started.
     * It is necessary if last time the processing of
     * current file has been interrupted
     * @param position the starting position
     */
    void setStartPosition(long position);
    /**
     * Returns the byte array that contains a portion of an incoming
     * file that most recently has been read and converted to Plain XML.
     * It should be in the same format as incoming file. The header and
     * tail records, if there are, should be included and updated according
     * to the content
     */
    byte[] getCurrentInputPortion();
    /**
     * Returns type of currently generated transaction - USAGE or EVENT
```

```
     */
    FileProcessor.TransactionType getCurrentTransactionType();
    /**
     * Returns the number of successfully generated transactions of given type
     */
    int getTransactionCount(FileProcessor.TransactionType type);
    /**
     * Returns the number of expected transactions in input stream
     * Returns -1 if information not available
     */
    int getTotalExpected();
}
```

FileParser2 Interface

This section provides the source code of the FileParser2 interface used by the Java class delivered with the Java package described earlier in this section. This interface has been added to include a way for the File Processor to retrieve the file creation date which can be stored inside the incoming file.

```
package com.splwg.d1.sgg.osb.common;
import java.util.Date;
public interface FileParser2 extends FileParser {
    /**
     * FileProcessor usage method
     * Returns the File creation date and time stamp stored
     * in the file
     * Returns null if information not available
     */
    Date getCreationDateTime();
}
```

FileProcessor Interface

This section contains a portion of the FileProcessor interface related to the file parsing functionality.

```
package com.splwg.d1.sgg.osb.common;
public interface FileProcessor {
    public enum TransactionType {
        USAGE, EVENT
    }
    /**
     * FileParser usage method
     *
     * @return a string containing the name of currently processed inbound file
     */
    String getOrigin();
    /**
     * FileParser usage method Creates a file in the Error Folder and saves
     * passed row parameter to it
     *
     * @param position
     *            the number representing the position in file where invalid
     *            raw data is started
     * @param row
     *            the byte array containing an invalid raw data in vendor
     *            specific format the same with format of incoming file
     * @param failCause
     *            the string showing the cause of fail It will be used while
     *            logging the fail message.
     */
    void saveInvalidRow(long position, byte[] row, String failCause);
    /**
     * FileParser usage method
     * Returns the current offset in input file
     */
    long getCurrentInputPosition();
```

```
}
```

Generating Java Classes from XML Schemas

In JDeveloper you can use JAXB (Java Architecture for XML Binding) to generate Java classes from XML schemas. JAXB is an easy way to incorporate XML data and processing functions in Java applications without having to know XML. You can generate a JAXB 1.0 or 2.0 content model, including the necessary annotations, from an XML schema.

When the JAXB binding compiler is run against an XML schema, JAXB packages, classes, and interfaces are generated. You can then use the generated JAXB packages and the JAXB utility packages in a binding framework to unmarshal, marshal, and validate XML content.

To generate Java classes from XML schemas with JAXB:

1. From the main menu choose **File > New > Business Tier > TopLink/JPA** and select either JAXB 1.0 or 2.0 Content Model from XML Schema to open the compilation dialog.
2. Select the schema file and optionally the JAXB customization file to use and the package to which the generated classes will be added.

The JAXB package and generated classes are added to the Application Resources folder.

Business Processing Execution Language Processing

This section describes the Business Process Execution Language (BPEL) processes included in the Oracle Utilities Smart Grid Gateway Adapter Development Kit that support two-way communication between the adapter and the head-end system. The ideas and general practices described in this section can be applied to any other adapter.

WSDLs, Endpoints, and Messages

WSDLs are Web Service Definition Language files that describe a web service. New services must utilize the WSDLs to determine the structure of requests and responses. For remote services, they contain the locations to access the services. For both remote and local (hosted) services, the files will contain the definitions for each web service including names, arguments, exceptions, and the structure of the input and output messages.

In Oracle Serivce Oriented Architecture (SOA), WSDLs can be classified as a “service” or as a “reference.” Services are hosted on the SOA server; that is, the implementation of the web service is on the local application server. Referenced services are implemented on a different server. This distinction is also relevant to the composite level. Services are implemented in the current composite. References are located elsewhere, possibly on the same app server, but in a different composite.

An “endpoint” is simply the URL for a web service. Since web services communicate via HTTP, each will have a unique URL. Once a service’s endpoint is known, a message service can be targeted to it.

How to locate WSDLs and Endpoints

There are two ways to locate WSDLs in an installed Adapter: through Enterprise Manager and by using a direct URL. Only “hosted” WSDLs can be located in this way. Referenced (remote) WSDLs must be either located in source code or obtained from the hosted location.

How to Use Enterprise Manager to Locate WSDLs and Endpoints:

1. Open Enterprise Manager and use the navigation pane to open the Dashboard of the desired composite.
2. The top bar of the Dashboard contains several buttons and icons. One of these is an “earth” icon with a puzzle piece over it. Click this icon to display the WSDL and endpoint URIs for the composite.

3. Click the URL link to see the WSDL in the browser, or right click and save it to your machine for use in developing new services.

   Depending on the requirements, It may be necessary to download the associated schema or WSDL file(s). Schemas are available within the WSDL's "types" element. Find associated WSDL URLs within the import element. That URL can be pasted into a browser tab.

   The endpoint for the service is also visible in this window. It is this URL that should be added to message sender configuration. For example:

   The value HTTP Header/SOA Action is taken from the service's WSDL. Locate the wsdl:definitions/wsdl:binding/wsdl:operation/soap:operation element for the web method being invoked. An attribute called soapAction will contain the value for this field. A shortcut to this field is using Enterprise Manager's Testing framework (locate the Test tab from the dashboard view for the composite). Once the WSDL is parsed, the SOAP Action will appear in a field on the request tab:

   The values **HTTP Login User** and **HTTP Login Password** should be set to a valid WebLogic user that has access to the module. The **HTTP Method** should always be set to POST and the HTTP URL 1 should be set to the value of the endpoint above.

How to Use a Direct URL to Locate WSDLs and Endpoints

The WSDLs can be accessed without Enterprise Manager by understanding the paths used on the SOA server. In general, they have the form:

```
http://{server name}:{port number}/soa-infra/services/{partition}/{Composite}/{Web Service}
```

Composite Components

This section outlines several important logical features of the BPEL composites.

Composite Properties

Most composites contain properties within the main file, composite.xml. They are essentially global constants that can be preconfigured with default values and changed at deployment time. They can be accessed post-deployment in Enterprise Manager.

Typically, these properties represent timeouts and boolean properties for activating or deactivating functionality. Other uses might include setting default values of constants such as URLs. They are accessed with the ora:getPreference() BPEL function.

They can be changed during development, at deployment time, or after deployment using different techniques for each.

Development Changes

At development time, properties can be created at will within the "component" sections of the composite.xml file. The image below shows an example of the properties found in DeviceStatusCheck:

```
<component name="DeviceStatusCheck">
  <implementation.bpel src="DeviceStatusCheck.bpel"/>
  <property name="bpel.preference.isExecutingInitODRequestReceivedCallout">true</property>
  <property name="bpel.preference.isExecutingInitODEventRequestCompleted">true</property>
  <property name="bpel.preference.isExecutingODEventNotificationArrived">true</property>
  <property name="bpel.preference.timeout.callback.years">0</property>
  <property name="bpel.preference.timeout.callback.months">0</property>
  <property name="bpel.preference.timeout.callback.days">0</property>
  <property name="bpel.preference.timeout.callback.hours">0</property>
  <property name="bpel.preference.timeout.callback.minutes">0</property>
  <property name="bpel.preference.timeout.callback.seconds">45</property>
  <property name="bpel.preference.OD_ServerCallbackEndpoint">http://127.0.0.1:8001/soa-infra/serv
  <property name="bpel.preference.externalSenderID">DSC Sensus</property>
  <property name="partnerLink.OD_ServerProxy.idempotent" type="xs:string"
            many="false">false</property>
</component>
```

Properties must be prefixed with “bpel.preference.” The BPEL process associated with them (see the implementation.bpel element) can use the ora:getPreference() method to extract the value. When accessing the values, the “bpel.preference” prefix is dropped.

Pre-Deployment Changes

While the values of properties cannot be changed at runtime, they can be altered using the configuration plan each composite uses for deployment. Most configuration plans will contain multiple property elements within the “component” section. Modifying the value in the replace element and redeploying the composite will change the behavior of the property:

```
<component name="DeviceStatusCheck">
   <property name="bpel.preference.isExecutingInitODRequestReceivedCallout">
      <replace>true</replace>
   </property>
   <property name="bpel.preference.isExecutingInitODEventRequestCompleted">
      <replace>true</replace>
   </property>
   <property name="bpel.preference.isExecutingODEventNotificationArrived">
      <replace>true</replace>
   </property>
   <property name="bpel.preference.timeout.callback.years">
      <replace>0</replace>
   </property>
   <property name="bpel.preference.timeout.callback.months">
      <replace>0</replace>
   </property>
   <property name="bpel.preference.timeout.callback.days">
      <replace>0</replace>
   </property>
```

When the composite is deployed, the properties will contain the values in the “replace” elements.

Post-Deployment Changes

After deployment, changes can still be made to the values of the properties through Enterprise Manager.

1. In the Enterprise Manager navigation window, open WebLogic Domain/{domain name}. Right click on the domain and select **System MBean Browser**.
2. A new navigation pane opens in the window on the right. Select **Application Defined MBeans/oracle.soa.config/Server: {domain name}/SCAComposite/{Composite} [1.0]/SCAComposite.SCAComponent/{Component}** to access the controls for a particular component. A “component” is a single BPEL or Mediator within a larger SOA composite

   A list of options will open in the right pane.

3. Select **Properties.**

   The Property sheet shows a list of all the properties. Opening one allows editing of the property value.

4. Click **Apply** to save any changes.

Proxies

Proxy web services make no changes to the data going through the system and are just used to centralize endpoint configuration. The message structures are identical to those of the head-end system. Data is passed through without modification. Mediator objects within the Common composite are used to move data from Oracle Utilities Smart Grid Gateway to the head-end system.

Process Callouts

Some services are designated as “Process Callouts.” These services allow integrators direct access to the XML coming to and from the head-end system. They are useful in the event the integrator needs to provide custom enrichment or transformation of the inbound or outbound data. Because process callouts are generally located before and after calls to the head-end system or when data arrives, they can also serve as an “event” trigger if the integrator requires additional processing logic. The services define identical inputs and outputs and are based on head-end schema definitions. To use a callout, the integrator should create a web service implementing one or more of the WSDLs defined in the Common composite. Other composites should then be configured to target the new Callout-type service.

A typical example is for re-rendering the meter identifier being sent to the head-end system. If the standard meter ID should be modified, a process callout service is a candidate for making this change. Note that because the schema is based off the head-end system's definition, deviations from the schema are not allowed.

Configuring and Customizing Adapter BPEL Processes

This section outlines how to configure and customize the Oracle Business Process Execution Language (BPEL) processes provided with the Adapter Development Kit to work with your implementation.

Editing Configuration Files

This section outlines the changes that need to made in various configuration and build files to enable the application servers used in your implementation to communicate with each other, and to support the specifics of your head-end system. These files are also used in packaging and deploying the processes.

Server Definitions

In order for your adapter to work properly, you must set up the application servers that run the components used by the adapter communicate with each other as follows:

- The Oracle Utilities Smart Grid Gateway application server must be able to send and receive messages to and from the SOA Suite application server.
- The SOA Suite application server must be able to send and receive messages to and from the head-end system application server, and the Oracle Utilities Smart Grid Gateway application server
- The head-end system application server must be able to receive and send messages from and to the SOA Suite application server.

The types of servers and the tokens that need to be replaced for each are listed below. Further below are the specific configuration files that must be modified, as well as the locations in each files that must be modified. Note that a port is listed, but may not be necessary depending on the type of installation. Also note that the SOA Server will have to have the partition name defined that is used.

- **SOA Server**: This is the application server running the SOA Suite components, including Oracle Service Bus (OSB) and Oracle Business Process Execution Language (BPEL). The SOA server is referenced using the following tokens:
  - {SOA_HOST}: the server where the SOA server has been installed
  - {SOA_PORT_NUMBER}: the port used by the SOA server
  - {SOA_PARTITION_DG}: the partition used by the SOA server. There are different partitions for the different adapters used by Oracle Utilities Smart Grid Gateway.
- **XAI (OUAF) Server**: This is the application server running the Oracle Utilities Smart Grid Gateway (including the Oracle Utilities Application Framework, or OUAF) software, including the inbound/outbound message components. This server is referenced using the following tokens:
  - {WEB_WLHOST}: the server where Oracle Utilities Smart Grid Gateway and the message components have been installed
  - {WEB_WLPORT}: the port used by the Oracle Utilities Smart Grid Gateway application
  - {WEB_CONTEXT_ROOT}: the root directory at which the Oracle Utilities Smart Grid Gateway inbound web service
- AMI Head-end Server: This is the application server running the head-end system software. The AMI head-end server is referenced using the following tokens:
  - {Headend_MR_Server_DG}: the URL for either the actual head-end system's MR_Server MultiSpeak implementation, or the URL to an emulator test harness being used (if applicable). For example, the Adapter Development Kit includes a soapUI configuration that can be used to emulate a head-end system for testing purposes.
  - {Headend_CD_Server_DG}: the URL for either the actual head-end system's CD_Server MultiSpeak implementation, or the URL to an emulator test harness being used (if applicable). For example, the generic adapter includes a soapUI configuration that can be used to emulate a head-end system for testing purposes.
  - Headend_OD_Server_DG}: the URL for either the actual head-end system's MR_Server MultiSpeak implementation, or the URL to an emulator test harness being used (if applicable). For example, the generic adapter includes a soapUI configuration that can be used to emulate a head-end system for testing purposes.

In addition, credentials for the WebLogic server must be specified in the build.properties file in the deploy folder:

- {WebLogic_UserID}: the user ID used to connect to the WebLogic server
- {WebLogic_Password}: the password for the {WebLogic_UserID} used to connect to the WebLogic server

Customizing BPEL Processes

This section provides an overview of how the sample BPEL processes provided with the Oracle Utilities Smart Grid Gateway Adapter Development Kit can be customized to meet specific business requirements, such as using custom XSL transformations and adding/editing the steps involved in the process.

Using Custom XSL Files

This process references XSL files to transform the messages from the Oracle Utilities Smart Gird Gateway standard format into the format used by the “generic” head-end system (based on the Multispeak protocol) when sending messages to the head-end system, and from the head-end system format into the Oracle Utilities Smart Gird Gateway standard format when sending messages back to Oracle Utilities Smart Gird Gateway. These XSL file references can be changed to custom XSL files that transform the messages into and from the format used by your head-end system. Refer to Oracle BPEL and Oracle JDeveloper documentation for more information about referencing XSL files within a BPEL process.

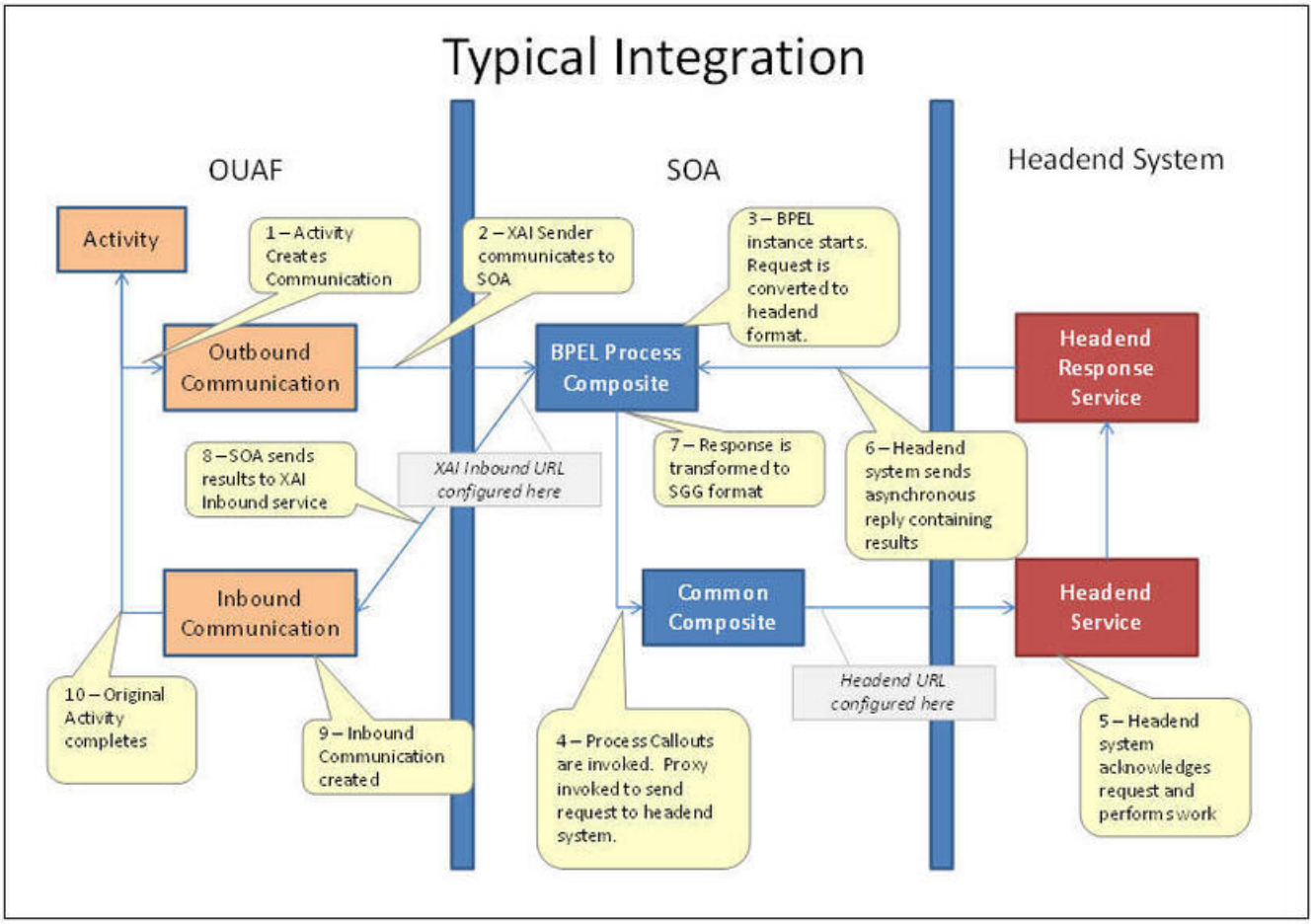
Making Changes to the Process

In addition to referencing custom XSL files for transformation, you can also add additional steps to the process to meet the specific business requirements of your implementation, or make other changes to the existing steps. Refer to Oracle BPEL and Oracle JDeveloper documentation for more information about adding and/or changing steps within BPEL process.

Note: Any changes made to this process for a particular AMI vendor and head-end system needs to be modeled based on the head-end system/AMI vendor's business processes, including configuring the required request/response transactions required for a specific command (such a remote connect).

Commands

This section describes the general structure and flow of each SOA composite in the adapter development kit for the two-way commands. Most follow this basic model:



Common

The Common composite is a repository for files and logic that is used by other composites. For instance, the Proxies and default implementations of Process Callouts are defined in the Common composite.

Composite Properties

None.

Composite WSDLs

Callout Services

Process Callouts are customization points for integrators with identical inputs and outputs based on head-end formats. The Common composite contains default implementations in which incoming data is reflected back in an identical state. Each of the following is a Mediator component and uses an “echo”-type definition.

Process Callouts are usually associated with boolean composite properties which control whether they will be executed. They are activated by default so uses can easily inject their customizations, but setting the properties to false can be a performance optimization. When set to false, the associated Process Callout will not be made and execution will continue normally.

Service	Description
OA_CalloutService	Used when the asynchronous reply in DeviceStatusCheck arrives.
MR_CalloutService	Used during Commission/Decommission and OnDemandRead operations.
CB_CalloutService	Used by the asynchronous callback processes of OnDemandRead and Connect/Disconnect.
CD_CalloutService	Used by Connect/Disconnect.
OD_CalloutService	Used in DeviceStatusCheck.

Proxies

Placing all the proxy web services in the Common composite creates a place to conveniently set endpoints. Each is a Mediator component which passes the data to the head-end system without modification

Proxy Web Service	Description
OD_ServerProxy	Used to transmit a DeviceStatusCheck request to the head-end system.
CD_ServerProxy	Used to transmit a Connect/Disconnect request to the head-end system.
MR_ServerProxy	Used to transmit an OnDemandRead request to the head-end system.

Other

AuxiliaryRoutinesService: A container defining helpful, commonly used functions:

- **FindExpTime:** Many MultiSpeak functions contain an expTime element, which is used to deliver the length of time to wait for the command to complete before failing. (The included test harness does not support timeouts.) If the input to the command composite does not contain this timeout, the process will read from its properties the years, months, days, hours, minutes, and seconds to wait for a result. This web service utilizes a Java method to combine these inputs into the proper XML field used in the MultiSpeak API. The properties are also used to control the timeout wait period of asynchronous callbacks within a command's BPEL process.
- **FindTimeout:** When the expTime element and units are supplied as input, it is usually still necessary to compute the timeout used in asynchronous callbacks. This method accepts the MultiSpeak fields and replies in the “P0Y0M0DT0H0M0S” format understood by BPEL.

Commission / Decommission

The CommissionDecommission composite takes care of registering and de-registering a device with the head-end system.

Composite Properties

Property Name	Default Value	Description
isExecutingCommissionReceived-Callout	true	Controls whether the Request Arrived callout executes.
isExecutingCommissionCompleted-Callout	true	Controls whether the Request Completed callout executes.
isExecutingDecommissionReceived-Callout	true	Controls whether the Request Arrived callout executes.
isExecutingDecommissionCompleted-Callout	true	Controls whether the Request Completed callout executes.

Composite WSDLs

Composite	Description
CommissionDecommission Service	Entry point for the CommissionDecommission command. The operations are synchronous, so no additional WSDLs are needed.
MR_ServerCallout	Reference describing the process callout.
MR_ServerProxy	Reference describing the proxy used to invoke the head-end system.

Process Flow

1. SGG/OUAF invokes MeterAddNotification/MeterRemoveNotification operation using CommissionDecommissionService.
2. Test whether MeterAddNotification or MeterRemoveNotification is invoked. In both cases, the following steps are common for both the operations.
3. Composite properties are loaded and local variables are initialized.
4. Both the Header and Body inputs are transformed to MultiSpeak format.
5. If required, execute a process callout in MR_ServerCallout.Assign updated data to head-end request.
6. Invoke MeterAddNotification/MeterRemoveNotification via the MR_ServerProxy to the head-end system.
7. If required, execute a process callout in MR_ServerCallout. Assign updated data to head-end response.
8. Transform Header and Body head-end responses to SGG/OUAF format.
9. Reply to SGG/OUAF with synchronous results.

Connect / Disconnect

The ConnectDisconnect composite is responsible for starting and stopping the recording of usage data for a meter.

Composite Properties

Property Name	Default Value	Description
CB_CDCallbackEndpoint	http://txdev2k3vm5.us.oracle.com:8001/soa-infra/services/DG/ConnectDisconnect/CB_ServerService	The default endpoint to which MultiSpeak should send the

Property Name	Default Value	Description
		asynchronous CB_Server callback.
isExecutingCDReceivedCallout	true	Controls whether the Request Received callout executes.
isExecutingCDCompletedCallout	true	Controls whether the Request Received Completed callout executes.
isExecutingCDStatesChangedArrivedCallout	true	Controls whether the CD States Changed Notification arrival callout executes.
CallbackTimeoutYears	0	The number of years to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutMonths	0	The number of months to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutDays	0	The number of days to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutHours	0	The number of hours to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutMinutes	0	The number of minutes to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutSeconds	45	The number of seconds to await a response from the MultiSpeak server. This property only takes effect when the expTime element is

Property Name	Default Value	Description
		not supplied in the input to the service.

Composite WSDLs

Composite	Description
ConnectDisconnectService	Entry point for the ConnectDisconnect operation. After a synchronous reply, further updates will be sent to the specified location asynchronously using DG-ConnectDisconnectStateChgNotification.
CB_ServerService	MultiSpeak WSDL hosted to receive asynchronous CDStatesChangedNotification callbacks from the head end system.
CB_ServerCallout	Reference describing the process callout used when the asynchronous callback from the head end system arrives.
CD_ServerCallout	Reference describing the process callout.
CD_ServerProxy	Reference describing the proxy used to invoke the head end system.
AuxiliaryRoutines	Reference to common helper routines in the Common composite.
DG-ConnectDisconnectStateChgNotification	A reference to an inbound web service capable of processing asynchronous results from a Connect/Disconnect request.

Process Flow

1. SGG/OUAF invokes InitiateConnectDisconnect operation using ConnectDisconnectService.
2. Composite properties are loaded and local variables are initialized.
3. Test whether the input includes a expiration time
   - If Yes: Use AuxilaryRoutines to compute timeout.
   - If No: Use composite properties and AuxiliaryRoutines to compute expiration time.
4. Both the Header and Body inputs are transformed to MultiSpeak format.
5. If required, execute a process callout in CD_ServerCallout. Assign updated data to head end request.
6. Add a callback URL to the head end request. This is the endpoint the head end will use to send the results.
7. Invoke InitiateConnectDisconnect via the CD_ServerProxy to the head end system.
8. If required, execute a process callout in CD_ServerCallout. Assign updated data to head end response.
9. Transform Header and Body head end responses to SGG/OUAF format.
10. Reply to SGG/OUAF with synchronous results.
11. Check for ErrorObject in the head end response.
12. If error object not found:

    Await asynchronous response from head-end system for the period of expiration time calculated during start of this flow:

    - Receive CDStatesChangeNotification from CB_ServiceService.
    - Generate reply with no errors for CDStatesChangeNotification
    - Synchronously reply to the head end.

- If required, execute a process callout in CB_ServerCallout. Assign updated data to the incoming request.
- Transform Header and Body of incoming request to OUAF inbound web service format described in DG-ConnectDisconnectStateChgNotification.
- If request contains responseURL
  - Invoke the inbound web service on given responseURL with the transformed incoming request.
- Else
  - Invoke the inbound web service on definedURL with the transformed incoming request.

Device Status Check

DeviceStatusCheck is used to determine the health of a meter and to test that the meter can be reached on the network.

Composite Properties

Property Name	Default Value	Description
OD_ServerCallbackEndpoint	http://127.0.0.1:8001/soa-infra/services/DG/DeviceStatusCheck/OA_ServerService	The default endpoint to which MultiSpeak should send the asynchronous OA_Server callback.
isExecutingInitODRequest-ReceivedCallout	true	Controls whether the Request Arrived callout executes.
isExecutingInitODEventRequest-Completed	true	Controls whether the Request Completed callout executes.
isExecutingODEventNotification-Arrived	true	Controls whether the ODEventNotification arrival callout executes.
timeout.callback.years	0	The number of years to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.months	0	The number of months to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.days	0	The number of days to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.hours	0	The number of hours to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.minutes	0	The number of minutes to await a response from the MultiSpeak server.

Property Name	Default Value	Description
		This property only takes effect when the expTime element is not supplied in the input to the service.
timeout.callback.seconds	45	The number of seconds to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Composite WSDLs

Composite	Description
OD_ServerService	Entry point for DeviceStatusCheck. After a synchronous reply, further updates will be sent to the specified location asynchronously using DG-OutageDetectionEventNotification.
OA_ServerService	MultiSpeak WSDL hosted to received asynchronous ODEventNotification callbacks from the head end system.
OD_ServerProxy	Reference describing the proxy used to invoke the head end system to initiate a Device Status Check.
OA_ServerCallout	Reference describing the process callout used when the asynchronous callback from the head end system arrives.
OD_ServerCallout	Reference describing the process callout used when initiating a Device Status Check.
AuxilaryRoutines	Reference to common helper routines in the Common composite.
DG-OutageDetectionEventNotificationXAI	A reference to an inbound web service capable of processing asynchronous results from a Device Status Check request.

Process Flow

1. SGG/OUAF Initiates the Outage Detection service using OD_ServerService
2. Composite properties are loaded. Local variables are initialized. These mostly include boolean settings describing the state (such as whether or not the synchronous reply from the head-end system returned)
3. Test whether the input includes a expiration time (expTime element).
   - If Yes: Use AuxilaryRoutines to compute timeout
   - If No: Use composite properties and AuxiliaryRoutines to compute expiration time
4. Both the Header and Body inputs are transformed to MultiSpeak format. This can be done either through direct mapping or through XSLT. In this instance, XSLT is used to transform the header and the body separately.
5. If the composite property indicates the process callout should occur, execute the “request arrived” process callout in OD_CalloutService. Assign updated data to head end request.
6. Add a callback URL to the head end request. This is the endpoint the head end will use to send the outage detection results. Some head end systems use other approaches, such as pre-configuring the URL for callbacks or allowing polling to find the results. In this case, the URL is configured in a composite property and corresponds to an implementation of the OA_Server WSDL. When the head end system has results, it will deliver them to this URL using the ODEventNotification method.

7. Invoke InitiateOutageDetectionEventRequest via the OD_ServerProxy to the head end system.
8. Split processing to handle simultaneous activities. The Flow activity does this in BPEL and it makes sense here because one process will handle a synchronous response and return it to the caller. The other process will sleep, or "dehydrate," until the asynchronous ODEventNotification arrives or until the timeout value is reached.
   - Handle synchronous response to SGG/OUAF
     - If the composite property indicates the process callout should occur, execute the "request completed" process callout in OD_CalloutService. Assign updated data to head end response.
     - Transform Header and Body head end responses to SGG/OUAF format.
     - Reply to SGG/OUAF with synchronous results.
     - Set boolean variable indicating that the synchronous reply has been returned.
   - Await asynchronous response from head-end system.
     - Receive ODEventNotification from OA_ServerService. If the timeout period elapses, raise an error and stop waiting.
     - Synchronously reply to the head end with no errors
     - If the composite property indicates the process callout should occur, execute the "data arrived" process callout in OA_CalloutService. Assign updated data to the incoming request.
     - Transform Header and Body to inbound web service format described in DG-OutageDetectionEventNotification.
     - Invoke the inbound web service with the new data.
9. Fault Handlers look for any remote Fault returned from the head-end system. The MultiSpeak API does not define named faults, but they are still possible to encounter. For example, a security error or network error could be raised.
   - If the synchronous reply has been sent back to the caller, Terminate the flow. This indicates an error has occurred and more research in Enterprise Manager is warranted.
   - If the synchronous reply has not been sent, abort further processing and compose a reply which contains the fault information.

On Demand Read

OnDemandRead interrogates a meter for the usage at the current point in time.

Composite Properties

Property Name	Default Value	Description
CB_ServerCallbackEndpoint	http://txdev2k3vm5.us.oracle.com:8001/soa-infra/services/DG/OnDemandRead/CB_Server	The default endpoint to which MultiSpeak should send the asynchronous CB_Server callback
IsExecutingOnDemandReadRequestReceived	true	Controls whether the Request Received callout executes
IsExecutingOnDemandReadRequestReceivedResponse	true	Controls whether the Request Received Completed callout executes
IsExecutingReadingChangedNotification	true	Controls whether the Reading Changed Notification arrival callout executes
CallbackTimeoutYears	0	The number of years to await a response from the MultiSpeak server. This property only takes effect when the

Property Name	Default Value	Description
		expTime element is not supplied in the input to the service.
CallbackTimeoutMonths	0	The number of months to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutDays	0	The number of days to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutHours	0	The number of hours to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutMinutes	0	The number of minutes to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.
CallbackTimeoutSeconds	45	The number of seconds to await a response from the MultiSpeak server. This property only takes effect when the expTime element is not supplied in the input to the service.

Composite WSDLs

Composite	Description
OnDemandReadService	Entry point for the OnDemandRead operation. After a synchronous reply, further updates will be sent to the specified location asynchronously using DG-ReadingChangedNotification.
CB_ServerService	MultiSpeak WSDL hosted to receive asynchronous ReadingChangedNotification callbacks from the head end system.
MR_ServerProxy	Reference describing the proxy used to invoke the head end system to initiate a On Demand Read.
MR_ServerCallout	Reference describing the process callout used when initiating a On Demand Read.
CB_ServerCallout	Reference describing the process callout used when the asynchronous callback from the head end system arrives.
AuxilaryRoutines	Reference to common helper routines in the Common composite.
DG-ReadingChangedNotificationXAI	A reference to an inbound web service capable of processing asynchronous results from a On Demand Read request.

Process Flow

1. SGG/OUAF invokes InitiateMeterReadingsByMeterID operation using OnDemandReadService.
2. Composite properties are loaded. Local variables are initialized.
3. Test whether the input includes a expiration time
   - If Yes: Use AuxilaryRoutines to compute timeout
   - If No: Use composite properties and AuxiliaryRoutines to compute expiration time
4. Both the Header and Body inputs are transformed to MultiSpeak format.
5. If required, execute a process callout in MR_ServerCallout. Assign updated data to head end request.
6. Add a callback URL to the head end request. This is the endpoint the head end will use to send the results.
7. Invoke InitiateMeterReadingsByMeterID via the MR_ServerProxy to the head end system.
8. If required, execute a process callout in MR_ServerCallout. Assign updated data to head end response.
9. Transform Header and Body head end responses to SGG/OUAF format
10. Reply to SGG/OUAF with synchronous results
11. Check for ErrorObject in the head end response.
12. If error object not found
    - Await asynchronous response from head end for the period of expiration time calculated during start of this flow
      - Receive ReadingChangedNotification from CB_ServiceService.
      - Generate reply with no errors for ReadingChangedNotification.
      - Synchronously reply to the head end.
      - If required, execute a process callout in CB_ServerCallout. Assign updated data to the incoming request.
      - Transform Header and Body of incoming request to OUAF inbound web service format described in DG-ReadingChangedNotificationXAI.
      - If request contains responseURL
        - Invoke the inbound web service on given responseURL with the transformed incoming request.
      - Else
        - Invoke the inbound web service on definedURL with the transformed incoming request.

Working with Enterprise Manager

Oracle Enterprise Manager (OEM) is useful for troubleshooting and diagnosing issues communicating with the head end system. In particular, security and other communications issues can be discerned.

To locate the instance of a service that is showing a problem, open the **Dashboard** view of the service. Each run instance is time-indexed. The **State** column in the first table contains the most important information for each instance. Completed messages show up with a green check icon. Terminated instances are indicated with a grey stop icon. Instances that are still running are marked as such. Runtime exceptions are in the bottom window.

When you click on the instance ID, a new window opens showing the entire process flow. This view is particularly useful in debugging.

Note the **State** column showing the status of each instance within the flow. SOA processes are made up of several calls to SOA components and web services. The **Instance** column shows a rough ordering of the operations. A typical approach to troubleshooting would find the “lowest” instance of an error or termination. This would be the error that is deepest within the process and is usually the source of the problem. In the above case, the deepest error is in the very first composite for the process. More information can be found by clicking into the link within the **Instance** column.

The level of detail is dependent on the auditing level set on the server. In this case, the error is in the response from the head end system.

OEM also contains a centralized location to control security. On a **Dashboard** screen for a composite, the **Policies** tab shows the OWSM policies attached to the composite. Typically, a log policy is placed on any inbound or outbound communication. Also, all services or references are delivered with attached basic http security policies.

MultiSpeak Implementation

This section lists the subset of MultiSpeak schemas and WSDLs utilized by the Adapter Development Kit.

MultiSpeak Head End System Service Definitions

- CB_Messages.wsdl
- CB_Server.wsdl
- CD_Messages.wsdl
- CD_PortTypes.wsdl
- CD_Server.wsdl
- MR_Messages.wsdl
- MR_PortTypes.wsdl
- MR_Server.wsdl
- OA_Messages.wsdl
- OA_Server.wsdl
- OA_ServerCollisions.wsdl
- OD_Messages.wsdl
- OD_PortTypes.wsdl
- OD_Server.wsdl

MultiSpeak Head End XML Schemas

- cpsm.xsd
- gml.xsd
- xlink.xsd
- mspcommon.xsd
- CB_Server.xsd
- CD_Server.xsd
- MR_Server.xsd
- OA_Server.xsd
- OD_Server.xsd

Configuring an Adapter Development Kit Head-End System

This section outlines the configuration required for the Oracle Utilities Smart Grid Gateway to communicate with the smart meters.

Inbound Web Services

Inbound web services define the details of how messages are received from an external system. This includes incoming usage and device events, as well as messages sent from the head-end system in response to a command request.

The following inbound web services must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating inbound web services.

Inbound Web Service Name	Description
D1-BulkRequestHeader	Bulk Request Header
D1-BulkRequestUpdate	Bulk Request Update
D1-BulkResponse	Bulk Response
D1-DeviceEventSeeder	Device Event Seeder
D1-DeviceStatusCheck	Device Status Check
D1-InitialLoadIMD	IMD Seeder
D1-PayloadErrorNotif	Payload Error Notification
D1-PayloadStatistics	Payload Statistics
D1-PayloadSummary	Payload Summary
DG-ConDisconStChgNotification	Initiate Connect Disconnect Response
DG-OutageDetectionEventNotification	Initiate Outage Detection Response
DG-ReadingChangedNotification	Reading Changed Notification

Note: The following apply to all of the above inbound web services:

Message Options

- **Trace**: No
- **Debug**: No
- **Active**: Yes

Operations

- **Operation Name**: Same as web service name
- **Schema Type**: Business Object
- **Schema Name**: Applicable business object code
- **Transaction Type**: Add

Message Senders

Message senders define the details of how messages are sent to an external system, such as messages containing device command requests.

The following message senders must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating message senders.

Message Sender	Description
DG-COMM	Generic Adapter Commission
DG-DCOMM	Generic Adapter Decommission
DG-INTOUTDET	Initiate Outage Detection Request
DGXAISender	Used for Meter Decommission

Note: The following apply to all of the above message senders:

Main Tab:

- **Invocation Type**: Real-time
- **Message Class**: RTHTTPSNDR (Sender routes message via HTTP real-time)
- **MSG Encoding**: UTF-8 message encoding

Context Tab:

- **HTTP Header**: SOAPAction: <**OPERATION**>
- **HTTP Login User**: <**USER_ID**>
- **HTTP Login Password**: <**PASSWORD**>
- **HTTP Method**: POST
- **HTTP URL 1**: http://<**EM_SERVER**>:<**EM_SERVER_PORT**>/soa-infra/services/Generic/<**SERVICE**>

  where:

  - <**OPERATION**>: the operation performed by the message sender (see Operation column in the table above)
  - <**USER_ID**>: the user ID used to log into WebLogic Enterprise Manager
  - <**PASSWORD**>: the password used to log into WebLogic Enterprise Manager
  - <**EM_SERVER_IP**>: the machine name or IP address of server where the WebLogic Enterprise Manager is installed
  - <**EM_SERVER_PORT**>: the port where the WebLogic Enterprise Manager is installed
  - <**SERVICE**>: the service invoked by the message sender (see Service column in the table above)

Outbound Message Types

Outbound message types define specific types of messages sent to an external system, such as messages containing device command requests.

The following outbound message types must be configured in your system. If these are not present in your configuration, add them. Refer to the Oracle Utilities Application Framework documentation for more information about creating outbound message types.

Outbound Message Type	Description
DG-COMM	Generic Adapter Commission
DG-DCOMM	Generic Adapter Decommission

Note: The following apply to all of the above outbound message types:

- **Business Object**: D1-OutboundMessage (Outbound Message)
- **Priority**: Priority 50

External System

External systems represent external applications with which the Smart Grid Gateway will exchange messages or data. In the case of the Smart Grid Gateway adapters, external systems represent the head-end systems with which the adapters communicate.

An external system that represents the head-end system must be present in your system. If this is not present in your configuration, add it, along with the following Outbound Message Types. Refer to the Oracle Utilities Application Framework documentation for more information about creating external systems.

External System — Generic:

- **External System**: Generic

- **Description**: Generic
- **Outbound Message Types**:

Outbound Message Type	Description	Message Sender
DG-COMM	Generic Adapter Commission	DG-COMM
DG-DCOMM	Generic Adapter Decommission	DG-DCOMM

Note: The following apply to all of the above outbound message types:

- **Processing Method**: Real-time
- **Message XSL**: DG-Request xsl
- **Response XSL**: DG-Response.xsl

Service Provider

Service providers represent external entities that serve various roles relative to the application, including head-end systems, billing systems to which the application sends bill determinant data, market participants in a deregulated environment, outage management systems that receive meter event data from the application, or other parties that require or provide information to the system. The head-end systems that collect and send measurement data and meter events to the application are defined as service providers.

A service provider that represents the head-end system must be present in your system. If this is not present in your configuration, add it. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about creating service providers.

Service Provider - Generic:

- **Service Provider**: Generic
- **Description**: Generic
- **External Reference ID**: Generic
- **External System**: Generic
- **Our Name/ID in Their System**:
- **AMI Device ID Type**: Internal Meter Number
- **AMI Measuring Component ID Type**: Channel ID

Processing Methods

Processing methods define the format or means by which a service provider receives and/or sends data from and/or to the application, including bill determinants, usage data, or device events. Processing methods are also used to define how to create information internal to the application such as initial measurement data and device events. Processing methods can also be used to define how command requests are sent to the head-end system.

The following types of processing methods must be configured for the head-end system service provider. Refer to the Oracle Utilities Service and Measurement Data Foundation documentation for more information about configuring processing methods.

Initial Measurement Creation

Initial measurement creation processing methods define the business objects used to create initial measurements. The IMD Seeder inbound web service uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from the head-end system.

Device Event Mapping

Device event mapping processing methods define how head-end-specific device events are mapped to standard device event names. The Device Event Seeder inbound web service uses this processing method to determine which type of device event business object to instantiate when receiving device events from the head-end system.

UOM Translation

UOM translation processing methods define how head-end-specific unit of measure (UOM) codes are mapped to standard UOM codes. This processing method is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Commands

Command processing methods define how command requests are sent to a head-end system. More specifically, they define the type of outbound communication business object to create for each type of command, and the outbound message type to send to the head-end system.

The following types of command processing methods can be configured for the head-end system service provider, based on the requirements of each implementation using the "How to Create OB COMM/Send OB Message" processing method business object (D1–HowToCreateActivityOBComm).

Command	Processing Role	Default Business Object	Default Outbound Message Type
Device Commission	Device Registration	DG-MeterAddNotification	Commission
Device Decommission	Device Removal	DG-MeterRemoveNotification	Decommission
Device Status Check	Device Status Check	DG-InitiateOutageDetection	Get Status
On-Demand Read (Scalar)	On-Demand Read (Scalar)	DG-InitiateMeterByMeterId	Initiate Meter Read by Meter ID
Remote Connect	Remote Connect	DG-InitiateConnectDisconnect	Connect Device
Remote Disconnect	Remote Disconnect	DG-InitiateConnectDisconnect	Disconnect Device

Configuring Endpoint URIs

Part of configuring your adapter is configuring your BPEL composites to work with your head-end system by defining the appropriate Endpoint URIs for each of the commands.

The default approach to defining Endpoints URIs is redeployment or reinstallation or the BPEL composites. For example, by default, changing an adapter from using the test harness to a production environment using the actual head end system requires editing the appropriate installation menu options and redeploying the BPEL composites. The Endpoints URIs defined during installation and deployment for each adapter are listed in the **Smart Grid Gateway Installation and Configuration Worksheets** section of the *Oracle Utilities Smart Grid Gateway Installation Guide*.

You can also use an "Endpoint Override" Domain Value Map (DVM) to override Endpoints URIs defined during deployment and installation. This DVM allows defining specific keys that provide an alternate URL that will override the original installed value. Endpoint Override DVMs can be edited using the Oracle SOA Composer.

The Adapter Development Kit endpoint override DVM (DG–EndpointOverrides.dvm) uses a specific set of keys, each used for one or more commands. The table below lists the DVM keys available for the Landis+Gyr adapter and the command used with each:

DVM Key	Commands
MR_Server	Device Commissioning
	Device Decommissioning

DVM Key	Commands
	On-Demand Read (Scalar)
	On-Demand Read (Interval)
CD_Server	Remote Connect
	Remote Disconnect
OD_Server	Device Status Check

To define an override Endpoint URI for an adapter built from the Adapter Development Kit, use the following procedure:

1. Open the SOA Composer for your BPEL configuration.

   The URL for the SOA Composer is http://server:port/soa/composer.

2. Select the DG–EndpointOverrides.dvm in the left panel.

   - In Fusion Middleware v12.2.1, this is located under **Shared** in the **Deployment View**, or under **Domain Value Maps** in the **Types View**.
   - In Fusion Middleware v12.2.2, this is located under **Metadata** in the **Deployment View**.

   The DVM will open in the right panel. The **Description** field lists the available keys for the DVM (only the first key is shown, but you can scroll through the contents to view the list. The panel also displays a list of previously defined keys.

3. Click the **Create Session** button (above the left panel) to begin an editing session.

4. Click the **Add Domain Values** icon ("+") to add a new key.

   The **Add Domain Values** dialog opens.

5. Enter the appropriate values in the **Add Domain Values** dialog as follows:

   - **key**: The DVM key for the Endpoint URI you wish to define (see the table above).
   - **EndpointURI**: The override Endpoint URI.

6. Click **OK**.

   The new DVM value will appear in the list of keys.

7. Click the **Save** icon to save the DVM values.

8. Click the **Publish** button to activate all the changes in the editing session.

   Enter an optional note for the session in the **Publish Session** dialog.

   Changes take effect immediately upon publishing the session.

   Click **Discard** to discard your changes.

   Click **Exit** to exit your current session. Note that your session will still be open if you exit. Use **Discard** to end your session without making changes.

Configuring Adapter Development Kit Extendable Lookups

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Itron OpenWay.

This section outlines some of the extendable lookups that must be configured for use with a customized adapter. Refer to the Oracle Utilities Application Framework documentation for more information about working with extendable lookups.

Device Event Mapping

The Device Event Mapping extendable lookup is used to determine which type of device event business object to instantiate when receiving device events from the head-end system.

Each value defined for the Device Event Mapping extendable lookup should include the following:

- **Head-End System Event Name**: The event name used by the head-end system.
- **Description**: A description of the device event
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Standard Event Name**: The standard event name for device events of this type, from the “Standard Event Name” extendable lookup.

Head-End UOM Code to Standard UOM Mapping

Usage received from the head-end system may use utility-specific unit of measures (UOMs). These custom UOMs must be mapped to standard UOM codes. The head-end system UOM Code to Standard UOM Mapping extendable lookup is used to determine how to map head-end system UOM codes to standard UOM codes when receiving usage from the head-end system.

Each value defined for the head-end system UOM Code to Standard UOM Mapping extendable lookup should include the following:

- **Head-End Unit of Measure**: The unit of measure code used by the head-end system
- **Unit of Measure**: The unit of measure defined in the system.
- **Description**: A description of the unit of measure code.
- **Status**: The status of the lookup value (can be Active or Inactive)

Interval Status Code to Condition Mapping

Interval usage received from the head-end system can include interval status codes that indicate the status or condition of the interval value. These interval status codes must be mapped to standard condition codes in the system. The Generic Interval Status Code to Condition Mapping extendable lookup is used to determine how to map head-end system interval status codes to standard status codes when receiving usage from the head-end system.

Each value defined for the Generic Interval Status Code to Condition Mapping extendable lookup should include the following:

- **Interval Status**: The head-end system interval status code
- **Description**: A description of the interval status code.
- **Status**: The status of the lookup value (can be Active or Inactive)
- **Condition**: The condition code to which the interval status code is to be mapped, from the Measurement Condition extendable lookup.

Other Extendable Lookups

Business Object Name	Description
DG-CDReasonCodeLookup	ADK - Connect Disconnect Reason Code
DG-GenericTimeUnits	ADK - Time Units
DG-LoadActionCodeLookup	ADK - Load Action Code
DG-OutageEventTypeLookup	ADK - Outage Event Type
DG-ServiceTypeMappingLookup	ADK - Service Type Mapping

Using the Adapter Development Kit Test Harness

Oracle Utilities Smart Grid Gateway Adapter Development Kit includes a test harness that can be configured to simulate a general head-end system for testing the two-way commands. The test harness includes a BPEL composite, web services for standard meter functions, and an XML file that can be used to contain information for one or more meters. This section describes the test harness and its components.

Locating the WSDL for the Test Harness

Follow these procedures to locate the Adapter Development Kit test harness WSDL:

How to Use Enterprise Manager to Locate the WSDL

1. Open Enterprise Manager and use the navigation pane to open the dashboard of the test harness composite:
2. The top bar of the dashboard contains several buttons and icons. One of these is a “world” icon with a puzzle piece over it. Click this icon to display a list of the WSDLs and endpoint URIs for the composite:
3. Click the UtilService WSDL URL link to see the WSDL in the browser, or right click and save it to your machine

Depending on your requirements, it may be necessary to download the associated schema found in the wsdl:types section. The URL can be pasted into a browser tab and downloaded in the same manner as the WSDL. The main schema has imported schemas that may also be required.

How to Use a Direct URL to locate the WSDL

The WSDL can be accessed without Enterprise Manager by understanding the paths used on the SOA server. In general, they have the following form:

```
http://{server name}:{port number}/soa-infra/services/{partition}/{Composite}/{Web Service}?WSDL
```

So by default, the test harness WSDL can be found at

```
http://{server name}:{port number}/soa-infra/services/DG_Test/DGTestHarness/UtilService?WSDL
```

Web Services

This section describes the web services included in the Adapter Development Kit test harness BPEL composite.

General Services

This section describes the general services of the Adapter Development Kit test harness composite.

LoadMeterIndex

This web service loads the data store from the internal file. By default if the store is already in memory, it will *not* reload. This behavior can be overridden with the forceReload parameter.

Input: LoadMeterIndexInput

Part: payload

Element: LoadMeterIndexRequest

Parameter	Description
forceReload	A switch telling the system whether to reload the meter index from the configuration file. Default value is false.

Output: LoadMeterIndexOutput

Part: payload

Element: LoadMeterIndexResult

Parameter	Description
loaded	A boolean value for whether or not the index was reloaded from the configuration file

Fault: UtilityFault (see UtilityFault for more details).

ViewAuditTrail

This web service returns the audit log for the entire session.

Input: ViewAuditTrailInput

Part: payload

Element: ViewAuditTrailRequest

Parameters: This is an empty request. There are no parameters.

Output: ViewAuditTrailOutput

Part: payload

Element: ViewAuditTrailResult

This element is an entry consisting of a timestamp and an Operation. Each entry may have an associated meter object showing what changed.

Fault: UtilityFault (see UtilityFault for more details).

UtilityFault

Fault with similar mapping to SGG/OUAF faults:

Typically, the faultCode, faultString, faultActor, and detail/text elements will be populated.

Locate Meter Services

This section describes the locate meter web services of the Adapter Development Kit test harness composite.

FindMeters

This web service queries the data store for one or more meters. The difference between GetMeter and FindMeters is GetMeter can return at most one meter and it must match the provided ID exactly. GetMeter will throw an error if the ID is not found. FindMeters can return more than one meter (when using the regex) and will not throw an error when the ID does not match any of the meters in the index.

Input: FindMetersInput

Part: payload

Element: FindMetersRequest

Parameter	Description
id	The meter ID for which to search
isRegex	The provided id can be a regex value when this parameter is true. Hint: to search for all meters in the system, use ".*" for the ID.

Output: FindMetersOutput

Part: payload

Element: FindMetersResult

Zero or more meter objects can be returned from the search

Fault: See UtilityFault. Unlike other methods, FindMeters does not throw an exception if the meter is not found.

IsMeterDefined

This web service queries whether a particular meter is defined in the data store.

Input: IsMeterDefinedInput

Part: payload

Element: IsMeterDefinedRequest

Parameter	Description
id	The meter ID for which to search

Output: IsMeterDefinedOutput

Part: payload

Element: IsMeterDefinedResult

Whether or not the provided ID is part of the index.

Fault: See UtilityFault. Thrown when meter id is not found.

GetMeter

This web service returns all the attributes of a single meter from the in-memory data store. The difference between GetMeter and FindMeters is GetMeter can return at most one meter and it must match the provided ID exactly. GetMeter will throw an error if the ID is not found. FindMeters can return more than one meter (when using the regex) and will not throw an error when the ID does not match any of the meters in the index.

Input: GetMeterInput

Part: payload

Element: GetMeterRequest

Parameter	Description
id	The meter ID for which to search

Output: GetMeterOutput

Part: payload

Element: GetMeterResult

The meter object requested by the ID.

Fault: See UtilityFault. Thrown when meter id is not found.

Meter Administration Services

This section describes the meter administration services of the Adapter Development Kit test harness composite.

AddMeters

This web service adds a set of meters to the in-memory data store. This will not permanently add it to the control file.

Input: AddMetersInput

Part: payload

Element: AddMetersRequest

Parameter	Description
id	The identification code for the meter.
utility	An informational string.
serviceType	One of the valid ServiceType values (see schema). "Electric" is the only option at this time.
isCommissioned	Whether or not the meter is in a commissioned state.
loadActionCode	One of the possible LoadActionCode values used in Connect and Disconnect (see schema).
outageEventType	One of the possible OutageEventType values used in Device Status Check (see schema).
executionStatus	One of the possible ExecutionStates (see schema). These values control how the meter will respond to commands.
updateIfExisting	Whether or not to update the meter with the provided values if it already exists in the index.
Comment	An informational string describing the purpose of the meter.
Channels	A listing of unit of measures supported by this meter.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.

Output: AddMetersOutput

Part: payload

Element: AddMetersResult

Whether or not each meter was added to the index.

Fault: See UtilityFault.

RemoveMeter

This web service removes a meter from the in-memory data store. This will not permanently remove it from the control file.

Input: RemoveMeterInput

Part: payload

Element: RemoveMeterRequest

Parameter	Description
id	The ID for the meter to be removed.

Output: RemoveMeterOutput

Part: payload

Element: RemoveMeterResult

Whether or not the meter was removed from the index.

Fault: See UtilityFault. Thrown when meter id is not found.

AddMeterChannel

This web service adds a new channel to a single meter.

Input: AddMeterChannelInput

Part: payload

Element: AddMeterChannelRequest

Parameter	Description
id	The identification code for the meter.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.

Output: AddMeterChannelOutput

Part: payload

Element: AddMeterChannelResult

Whether or not the channel was added to the index.

Fault: See UtilityFault. Thrown when meter id is not found.

RemoveMeterChannel

This web service removes a Channel from a meter.

Input: RemoveMeterChannelInput

Part: payload

Element: RemoveMeterChannelRequest

Parameter	Description
id	The ID for the meter to be removed.
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.

These three parameters are combined to locate a unique channel

Output: RemoveMeterChannelOutput

Part: payload

Element: RemoveMeterChannelResult

Whether or not the channel was removed from the meter.

Fault: See UtilityFault. Thrown when meter id is not found.

ReadScalarMeter

This web service generates a scalar reading for each channel of a given meter.

Input: ReadScalarMeterInput

Part: payload

Element: ReadScalarMeterRequest

Parameter	Description
id	The ID for the meter to be read.

Output: ReadScalarMeterOutput

Part: payload

Element: ReadScalarMeterResult

Zero or more scalar readings for the given meter.

Parameter	Description
uomCode	A code describing the unit of measure for the channel.
uomName	A short string containing the name of the unit of measure.
decimals	The number of digits to the right of the decimal that should be generated when reading the meter.
description	A longer description of the unit of measure.
value	A random number representing the scalar reading.

Fault: See UtilityFault. Thrown when meter id is not found.

Meter Attribute Administration Services

This section describes the meter administration services of the Adapter Development Kit test harness composite.

GetOutageEventType

This web service queries the outage event type for a given meter. The OutageEventType is used by DeviceStatusCheck.

Input: GetOutageEventTypeInput

Part: payload

Element: GetOutageEventTypeRequest

Parameter	Description
id	The ID for the meter for which the OutageEventType should be retrieved.

Output: GetOutageEventTypeOutput

Part: payload

Element: GetOutageEventTypeResult

The value of the OutageEventType attribute for the requested meter.

Fault: See UtilityFault. Thrown when meter id is not found.

SetOutageEventType

This web service updates the outage event type for a given meter.

Input: SetOutageEventTypeInput

Part: payload

Element: SetOutageEventTypeRequest

Parameter	Description
id	The ID for the meter for which the OutageEventType should be set.
value	The new value of OutageEventType to set on the meter.

Output: SetOutageEventTypeOutput

Part: payload

Element: SetOutageEventTypeResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault: See UtilityFault. Thrown when meter id is not found.

GetLoadActionCode

This web service queries the load action code for a given meter. This is the Connect/Disconnect behavior.

Input: GetLoadActionCodeInput

Part: payload

Element: GetLoadActionCodeRequest

Parameter	Description
id	The ID for the meter for which the LoadActionCode should be retrieved.

Output: GetLoadActionCodeOutput

Part: payload

Element: GetLoadActionCodeResult

The value of the LoadActionCode attribute for the requested meter.

Fault: See UtilityFault. Thrown when meter id is not found.

SetLoadActionCode

This web service updates the load action code for a given meter.

Input: SetLoadActionCodeInput

Part: payload

Element: SetLoadActionCodeRequest

Parameter	Description
id	The ID for the meter for which the LoadActionCode should be set.
value	The new value of LoadActionCode to set on the meter.

Output: SetLoadActionCodeOutput

Part: payload

Element: SetLoadActionCodeResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault: See UtilityFault. Thrown when meter id is not found.

IsCommissioned

This web service queries the commissioning status for a given meter. This is the Commission/Decommission behavior.

Input: IsCommissionedInput

Part: payload

Element: IsCommissionedRequest

Parameter	Description
id	The ID for the meter for which the Commissioned status should be retrieved.

Output: IsCommissionedOutput

Part: payload

Element: IsCommissionedResult

The value of the Commissioned status attribute for the requested meter.

Fault: See UtilityFault. Thrown when meter id is not found.

SetCommission

This web service updates the commissioning status for a given meter.

Input: SetCommissionedInput

Part: payload

Element: SetCommissionedRequest

Parameter	Description
id	The ID for the meter for which the Commissioned status should be set.
value	The new value of Commissioned status to set on the meter.

Output: SetCommissionedOutput

Part: payload

Element: SetCommissionedResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault: See UtilityFault. Thrown when meter id is not found.

GetExecutionStatus

This web service queries the status of the property controlling the overall execution of the command.

Input: GetExecutionStatusInput

Part: payload

Element: GetExecutionStatusRequest

Parameter	Description
id	The ID for the meter for which the ExecutionStatus should be retrieved.

Output: GetExecutionStatusOutput

Part: payload

Element: GetExecutionStatusResult

The value of the ExecutionStatus attribute for the requested meter.

Fault: See UtilityFault. Thrown when meter id is not found.

SetExecutionStatus

This web service updates the property controlling the overall completion of the command.

Input: SetExecutionStatusInput

Part: payload

Element: SetExecutionStatusRequest

Parameter	Description
id	The ID for the meter for which the ExecutionStatus should be set.
value	The new value of ExecutionStatus to set on the meter.

Output: SetExecutionStatusOutput

Part: payload

Element: SetExecutionStatusResult

The boolean response indicates the success or failure of the update (not the current field status).

Fault: See UtilityFault. Thrown when meter id is not found.

Sample Meters File

The Adapter Development Kit includes an XML file and schema that can be used for configuring one or more meters for use with the test harness. The file, metersdb.xml, is located in the Test/DGTestHarness directory and can be edited with an appropriate XML editor such as XML Spy. This section describes the attributes in the metersdb.xml file.

The meterdb.xml file contains one or more **Meter** elements that have the following attributes:

Meter Attribute	Definition
id	The meter identifier. This value should match the amiDeviceID setting in the Oracle Utilities Application Framework.
utility	This is informational only.
serviceType	This is informational only.

Meter Attribute	Definition
ServiceType	An enumeration that is described in the schema.
isCommissioned	A boolean value that describes whether or not the meter is commissioned or decommissioned. The associated MultiSpeak commands are MeterAddNotification and MeterRemoveNotification.
loadActionCode	This value is for the Connect and Disconnect commands and has enumeration values described in the schema. The value is returned in the CDStatesChangedNotification MultiSpeak command.
outageEventType	This value is used by DeviceStatusCheck and gives the status returned by the ODEventNotification call. Its enumeration values are described in the schema.
executionStatus	This value does not reflect a MultiSpeak command, but is instead intended to give a state of the operation. There are four valid values: **Success:** When a meter has this status the operation will complete without error. **ResponseTimeout:** When a meter has this status an asynchronous reply will never arrive (not relevant for Commission/Decommission commands). **SyncOperationFailure:** When a meter has this status the initial communication to the simulated head-end system will produce an error. **AsyncOperationFailure:** When a meter has this status, the asynchronous callback from the head-end system will arrive, but will indicate an error (not relevant for Commission/ Decommission commands).

A meter can also contain the following elements:

- **Comment:** A field which is for informational use only and is meant to indicate the purpose of the meter.
- **Channels:** Used in reading the meter for On Demand Read commands. A channel contains the following attributes:

Channel Attribute	Definition
uomCode	The units of measure code that should be returned when the meter is read.
uomName	The units of measure name that should be returned when the meter is read.
description	A longer form. When the meter is read, a random number is generated for this value.
decimals	A value to indicate how many places to the right of the decimal that random number should have.

The Adapter Development Kit Native Format

NOTE: This section applies to both cloud and on-premises implementations of the Smart Grid Gateway Adapter for Itron OpenWay.

The Adapter Development Kit supports loading usage and event data exported from the AMI head-end system in the "native" initial measurement and device event data formats (the format of the initial measurement and device event seeder business objects). Processing of the ADK native format is supported by the following OSB projects:

- **SGG-DG-SEEDER-BASE**
- **SGG-DG-SEEDER-CM**

Refer to Initial Measurements and Device Events and OSB Project Summary for more information about these OSB projects.

Format Details

The ADK native format is an XML format that contains zero or more initial measurements and/or device events, as follows:

- The collection of initial measurements and/or device events are encapsulated within in an <SGGIMDsEvents> element.
- Each initial measurement or device event is defined in the format of the initial measurement and device event seeder business objects, encapsulated by the following elements:

- **Initial Measurements**: <D1-InitialLoadIMD>
- **Device Events**: <D1–DeviceEventSeeder>

- The format can support any number of initial measurements and/or device events.
- Initial measurements can be either scalar or interval measurements.

See Adapter Development Kit Native Format Example for an example of this format. See Adapter Development Kit Native Format Schema for the native format XML schema.

Adapter Development Kit Native Format Example

The following is an example of the ADK native format. This example contains 2 initial measurements (1 scalar and 1 interval) and 2 device events (PowerOutage and PowerRestored).

```
<?xml version="1.0" encoding="UTF-8" ?>
<SGGIMDsEvents xmlns="http://oracle.com/SGGIMDsEvents">
  <D1-InitialLoadIMD dateTimeTagFormat="xsd" xmlns="">
    <preVEE>
      <imdType>D1IL</imdType>
      <mcIS>D1SC</mcIS>
      <externalId>da_164_Scalar_withMetadata.xml-2015-09-10-11-19-03-297</externalId>
      <dvcIdN>DL_06</dvcIdN>
      <externalUOM>KWH</externalUOM>
      <enDt>2005-01-01T00:00:00Z</enDt>
      <enQty>4544</enQty>
    </preVEE>
    <serviceProviderExternalId>Itron</serviceProviderExternalId>
  </D1-InitialLoadIMD>
  <D1-InitialLoadIMD dateTimeTagFormat="xsd" xmlns="">
    <preVEE>
      <imdType>D1IL</imdType>
      <mcIS>D1IN</mcIS>
      <externalId>da_164_Multile_withMetadata_A.xml-2015-09-10-11-19-03-297</externalId>
      <dvcIdN>ZZ-D-OSB-INT-ITRON-0002</dvcIdN>
      <spi>900</spi>
      <externalUOM>KWH</externalUOM>
      <mcm>725</mcm>
      <stDt>2003-01-01T23:45:00Z</stDt>
      <enDt>2003-01-02T00:00:00Z</enDt>
      <msrs>
        <mL>
          <s>1</s><q>4722</q>
          <sts>
            <stsL>
              <s>1</s>
              <st>CHardwareFailure</st>
            </stsL>
            <stsL>
              <s>2</s>
              <st>COverflow</st>
            </stsL>
          </sts>
        </mL>
      </msrs>
    </preVEE>
    <serviceProviderExternalId>Itron</serviceProviderExternalId>
  </D1-InitialLoadIMD>
  <D1-DeviceEventSeeder dateTimeTagFormat="xsd" xmlns="">
    <externalSenderId>Itron</externalSenderId>
        <deviceIdentifierNumber>ZZ-D-OSB-INT-ITRON-0002</deviceIdentifierNumber>
    <externalEventName>PowerOutage</externalEventName>
    <eventDateTime>2001-02-02T00:00:00Z</eventDateTime>
    <eventInformation>
      <externalEventIdentifier>2147483642</externalEventIdentifier>
      <externalStatusValue>string</externalStatusValue>
      <externalEventCategory>Communication</externalEventCategory>
```

```
    </eventInformation>
  </D1-DeviceEventSeeder>
  <D1-DeviceEventSeeder dateTimeTagFormat="xsd" xmlns="">
    <externalSenderId>Itron</externalSenderId>
    <externalSourceIdentifier>da_164_Multile_withMetadata_A.xml-2015-09-10-11-19-03-297</
externalSourceIdentifier>
    <deviceIdentifierNumber>ZZ-D-OSB-INT-ITRON-0002</deviceIdentifierNumber>
    <externalEventName>PowerRestored</externalEventName>
    <eventDateTime>2001-02-02T01:00:00Z</eventDateTime>
    <eventInformation>
      <externalEventIdentifier>2147483642</externalEventIdentifier>
      <externalStatusValue>string</externalStatusValue>
      <externalEventCategory>Communication</externalEventCategory>
    </eventInformation>
  </D1-DeviceEventSeeder>
</SGGIMDsEvents>
```

Adapter Development Kit Native Format Schema

The following is the XML schema of the ADK native format.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="SGGIMDsEvents">
<xsd:complexType>
      <xsd:all>
 <xsd:element name="D1-DeviceEventSeeder" minOccurs="0">
  <xsd:complexType>
   <xsd:all>
    <xsd:element name="deviceEventId" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="14" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="bo" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="30" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="boStatus" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="12" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="sender" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="30" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="externalSenderId" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="36" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="deviceEventType" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="30" />
      </xsd:restriction>
     </xsd:simpleType>
```

```
    </xsd:element>
    <xsd:element name="externalEventName" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="254" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="eventDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/>
    <xsd:element name="eventEndDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" />
    <xsd:element name="externalTimeZone" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="20" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="deviceId" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="12" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="deprecatedDeviceId" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="12" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="creationDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" />
    <xsd:element name="statusUpdateDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" />
    <xsd:element name="statusReason" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="30" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="rawEventInformation" type="xsd:anyType" minOccurs="0" />
    <xsd:element name="externalSourceIdentifier" minOccurs="0">
     <xsd:simpleType>
      <xsd:restriction base="xsd:string">
       <xsd:maxLength value="120" />
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="eventInformation" minOccurs="0">
     <xsd:complexType>
      <xsd:all>
       <xsd:element name="externalEventIdentifier" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalEventCategory" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalEventSeverity" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
```

```
       <xsd:element name="externalDeviceType" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalServiceLocationId" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalCommunicationModuleIdentifier" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalGatewayIdentifier" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalStatusValue" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalStatusDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" /
>
       <xsd:element name="externalCommandId" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalEventDescription" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="60" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalEventReason" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="30" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="externalStatusReason" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="60" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="sourceTimeZone" minOccurs="0">
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
          <xsd:maxLength value="10" />
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
```

```
      <xsd:element name="timeZone" minOccurs="0">
       <xsd:simpleType>
        <xsd:restriction base="xsd:string">
         <xsd:maxLength value="10" />
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:element>
      <xsd:element name="dateTimesInStandard" minOccurs="0">
       <xsd:simpleType>
        <xsd:restriction base="xsd:string">
         <xsd:enumeration value="D1NO" />
         <xsd:enumeration value="D1YS" />
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:element>
     </xsd:all>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="version" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="-99999" />
      <xsd:maxExclusive value="99999" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="deviceIdentifierNumber" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="60" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="newDeviceEvent" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="14" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="nextRetryDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" />
   <xsd:element name="retryUntilDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" />
   <xsd:element name="processData" minOccurs="0">
    <xsd:complexType>
     <xsd:all>
      <xsd:element name="errorEncountered" minOccurs="0">
       <xsd:simpleType>
        <xsd:restriction base="xsd:string">
         <xsd:enumeration value="D1NO" />
         <xsd:enumeration value="D1YS" />
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:element>
      <xsd:element name="dateTimesInStandard" minOccurs="0">
       <xsd:simpleType>
        <xsd:restriction base="xsd:string">
         <xsd:enumeration value="D1NO" />
         <xsd:enumeration value="D1YS" />
        </xsd:restriction>
       </xsd:simpleType>
      </xsd:element>
      <xsd:element name="logs" minOccurs="0">
       <xsd:complexType>
        <xsd:all>
         <xsd:element name="logsList" minOccurs="0" >
          <xsd:complexType>
           <xsd:all>
            <xsd:element name="logsEntry" minOccurs="0">
             <xsd:complexType>
              <xsd:all>
               <xsd:element name="sequence" minOccurs="0">
                <xsd:simpleType>
```

```
                  <xsd:restriction base="xsd:decimal">
                   <xsd:minExclusive value="-999" />
                   <xsd:maxExclusive value="999" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="mo" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="12" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="pkValue1" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="50" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="pkValue2" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="50" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="pkValue3" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="50" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="pkValue4" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="50" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="pkValue5" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="50" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="logEntryType" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:enumeration value="D1TD" />
                   <xsd:enumeration value="F1CR" />
                   <xsd:enumeration value="F1ER" />
                   <xsd:enumeration value="F1EX" />
                   <xsd:enumeration value="F1ST" />
                   <xsd:enumeration value="F1SY" />
                   <xsd:enumeration value="F1TD" />
                   <xsd:enumeration value="F1US" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="logDateTime" nillable="true" type="xsd:dateTime" minOccurs="0" /
>
                <xsd:element name="boStatus" minOccurs="0">
                 <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                   <xsd:maxLength value="12" />
                  </xsd:restriction>
                 </xsd:simpleType>
                </xsd:element>
                <xsd:element name="description" minOccurs="0">
```

```
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="60" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="user" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="8" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="logMessage" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="4000" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="characteristicType" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="20" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="characteristicValue" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="16" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="adhocValue" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="254" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="fkValue1" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="50" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="fkValue2" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="50" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="fkValue3" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="50" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="fkValue4" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="50" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="fkValue5" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
```

```
      <xsd:maxLength value="50" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageCategory" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="-99999" />
      <xsd:maxExclusive value="99999" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageNumber" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:decimal">
      <xsd:minExclusive value="-99999" />
      <xsd:maxExclusive value="99999" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm1" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm2" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm3" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm4" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm5" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm6" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm7" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="30" />
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="messageParm8" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
```

```
                  <xsd:maxLength value="30" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
               <xsd:element name="messageParm9" minOccurs="0">
                <xsd:simpleType>
                 <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="30" />
                 </xsd:restriction>
                </xsd:simpleType>
               </xsd:element>
              </xsd:all>
             </xsd:complexType>
            </xsd:element>
           </xsd:all>
            <xsd:attribute name="action" type="xsd:string" use="optional" />
          </xsd:complexType>
         </xsd:element>
        </xsd:all>
       </xsd:complexType>
      </xsd:element>
     </xsd:all>
    </xsd:complexType>
   </xsd:element>
  </xsd:all>
 </xsd:complexType>
</xsd:element>
 <xsd:element name="D1-InitialLoadIMD" minOccurs="0">
   <xsd:complexType>
     <xsd:all>
       <xsd:element name="initialMeasurementDataId" minOccurs="0">
         <xsd:simpleType>
           <xsd:restriction base="xsd:string">
             <xsd:maxLength value="14"/>
           </xsd:restriction>
         </xsd:simpleType>
       </xsd:element>
       <xsd:element name="bo" minOccurs="0">
         <xsd:simpleType>
           <xsd:restriction base="xsd:string">
             <xsd:maxLength value="30"/>
           </xsd:restriction>
         </xsd:simpleType>
       </xsd:element>
       <xsd:element name="fromDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/>
       <xsd:element name="toDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/>
       <xsd:element name="boStatus" minOccurs="0">
         <xsd:simpleType>
           <xsd:restriction base="xsd:string">
             <xsd:maxLength value="12"/>
           </xsd:restriction>
         </xsd:simpleType>
       </xsd:element>
       <xsd:element name="statusReason" minOccurs="0">
         <xsd:simpleType>
           <xsd:restriction base="xsd:string">
             <xsd:maxLength value="30"/>
           </xsd:restriction>
         </xsd:simpleType>
       </xsd:element>
       <xsd:element name="dataSource" minOccurs="0">
         <xsd:simpleType>
           <xsd:restriction base="xsd:string">
             <xsd:enumeration value="D1RE"/>
             <xsd:enumeration value="D1ST"/>
           </xsd:restriction>
         </xsd:simpleType>
       </xsd:element>
       <xsd:element name="timeZone" minOccurs="0">
         <xsd:simpleType>
           <xsd:restriction base="xsd:string">
             <xsd:maxLength value="10"/>
```

```
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="creationDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/>
        <xsd:element name="comments" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="254"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="isTraceOn" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="D1NO"/>
              <xsd:enumeration value="D1YS"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="isIntervalDateTimePopulated" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="D1NO"/>
              <xsd:enumeration value="D1YS"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="serviceProvider" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="30"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="isAutomatedRetry" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="D1NO"/>
              <xsd:enumeration value="D1YS"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="nextRetryDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/>
        <xsd:element name="retryUntilDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/
>
        <xsd:element name="veeGroupRole" nillable="true" minOccurs="0">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="D1ES"/>
              <xsd:enumeration value="D1IL"/>
              <xsd:enumeration value="D1IS"/>
              <xsd:enumeration value="D1MO"/>
              <xsd:enumeration value="D1PR"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="startReadingDateTime" type="xsd:dateTime" minOccurs="0"/>
        <xsd:element name="startReading" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:decimal">
              <xsd:minExclusive value="-9999999999.999999"/>
              <xsd:maxExclusive value="9999999999.999999"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="startReadingCondition" type="xsd:string" minOccurs="0"/>
        <xsd:element name="preVEE" minOccurs="0">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="simdId" minOccurs="0">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
```

```
            <xsd:maxLength value="14"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="dvcIdN" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="120"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="mcId" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="12"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="mcIdN" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="120"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="externalId" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="120"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="uom" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="externalUOM" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="tou" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="externalTOU" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="sqi" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="externalSQI" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
          </xsd:restriction>
```

```
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="stDt" nillable="true" type="xsd:dateTime" minOccurs="0"/>
    <xsd:element name="stQty" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
          <xsd:minExclusive value="-9999999999.999999"/>
          <xsd:maxExclusive value="9999999999.999999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="enDt" nillable="true" type="xsd:dateTime" minOccurs="0"/>
    <xsd:element name="enQty" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
          <xsd:minExclusive value="-9999999999.999999"/>
          <xsd:maxExclusive value="9999999999.999999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="imdType" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="D1ES"/>
          <xsd:enumeration value="D1GA"/>
          <xsd:enumeration value="D1IL"/>
          <xsd:enumeration value="D1MO"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="mcIS" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="D1IN"/>
          <xsd:enumeration value="D1SC"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="inShift" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="D1NS"/>
          <xsd:enumeration value="D1SH"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="mcm" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
          <xsd:minExclusive value="-999999999999.999999"/>
          <xsd:maxExclusive value="999999999999.999999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="nd" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
          <xsd:minExclusive value="-999999999"/>
          <xsd:maxExclusive value="999999999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="tz" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="10"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="externalTimeZone" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
```

```
                    <xsd:maxLength value="20"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
              <xsd:element name="spi" nillable="true" type="xsd:int" minOccurs="0"/>
              <xsd:element name="ccond" type="xsd:string" minOccurs="0"/>
              <xsd:element name="sts" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="stsL" minOccurs="0" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="s" minOccurs="0">
                            <xsd:simpleType>
                              <xsd:restriction base="xsd:decimal">
                                <xsd:minExclusive value="-99999"/>
                                <xsd:maxExclusive value="99999"/>
                              </xsd:restriction>
                            </xsd:simpleType>
                          </xsd:element>
                          <xsd:element name="st" minOccurs="0">
                            <xsd:simpleType>
                              <xsd:restriction base="xsd:string">
                                <xsd:maxLength value="6"/>
                              </xsd:restriction>
                            </xsd:simpleType>
                          </xsd:element>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="msrs" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="mL" minOccurs="0" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="s" minOccurs="0">
                            <xsd:simpleType>
                              <xsd:restriction base="xsd:decimal">
                                <xsd:minExclusive value="-99999"/>
                                <xsd:maxExclusive value="99999"/>
                              </xsd:restriction>
                            </xsd:simpleType>
                          </xsd:element>
                          <xsd:element name="dt" nillable="true" type="xsd:dateTime" minOccurs="0"/
>
                          <xsd:element name="q" minOccurs="0">
                            <xsd:simpleType>
                              <xsd:restriction base="xsd:decimal">
                                <xsd:minExclusive value="-9999999999.999999"/>
                                <xsd:maxExclusive value="9999999999.999999"/>
                              </xsd:restriction>
                            </xsd:simpleType>
                          </xsd:element>
                          <xsd:element name="ue" minOccurs="0">
                            <xsd:simpleType>
                              <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="D1UE"/>
                              </xsd:restriction>
                            </xsd:simpleType>
                          </xsd:element>
                          <xsd:element name="fc" type="xsd:string" minOccurs="0"/>
                          <xsd:element name="r" minOccurs="0">
                            <xsd:simpleType>
                              <xsd:restriction base="xsd:decimal">
                                <xsd:minExclusive value="-9999999999.999999"/>
                                <xsd:maxExclusive value="9999999999.999999"/>
                              </xsd:restriction>
                            </xsd:simpleType>
```

```
                        </xsd:element>
                        <xsd:element name="rc" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="sts" minOccurs="0">
                          <xsd:complexType>
                            <xsd:sequence>
                              <xsd:element name="stsL" minOccurs="0" maxOccurs="unbounded">
                                <xsd:complexType>
                                  <xsd:sequence>
                                    <xsd:element name="s" minOccurs="0">
                                      <xsd:simpleType>
                                        <xsd:restriction base="xsd:decimal">
                                          <xsd:minExclusive value="-99999"/>
                                          <xsd:maxExclusive value="99999"/>
                                        </xsd:restriction>
                                      </xsd:simpleType>
                                    </xsd:element>
                                    <xsd:element name="st" minOccurs="0">
                                      <xsd:simpleType>
                                        <xsd:restriction base="xsd:string">
                                          <xsd:maxLength value="6"/>
                                        </xsd:restriction>
                                      </xsd:simpleType>
                                    </xsd:element>
                                  </xsd:sequence>
                                </xsd:complexType>
                              </xsd:element>
                            </xsd:sequence>
                          </xsd:complexType>
                        </xsd:element>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="rawData" type="xsd:anyType" minOccurs="0"/>
      <xsd:element name="processData" minOccurs="0">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="isShiftedStartEnd" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="D1NO"/>
                  <xsd:enumeration value="D1YS"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="isShiftedIntervals" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="D1NO"/>
                  <xsd:enumeration value="D1YS"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="isTimeZoneConverted" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="D1NO"/>
                  <xsd:enumeration value="D1YS"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="isErrorEncountered" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="D1NO"/>
                  <xsd:enumeration value="D1YS"/>
                </xsd:restriction>
```

```
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="servicePointId" minOccurs="0">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:maxLength value="12"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="installationConstant" minOccurs="0">
            <xsd:simpleType>
              <xsd:restriction base="xsd:decimal">
                <xsd:minExclusive value="-999999.999999"/>
                <xsd:maxExclusive value="999999.999999"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="deviceId" minOccurs="0">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:maxLength value="12"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="allowNonZeroIntervalsForEstimate" minOccurs="0">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="D1NO"/>
                <xsd:enumeration value="D1YS"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="disableReEstimate" minOccurs="0">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="D1NO"/>
                <xsd:enumeration value="D1YS"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="reEstimationActivity" minOccurs="0">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:maxLength value="14"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="logs" minOccurs="0">
            <xsd:complexType>
              <xsd:all>
                <xsd:element name="logsList" minOccurs="0">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="logsEntry" minOccurs="0">
                        <xsd:complexType>
                          <xsd:sequence>
                            <xsd:element name="sequence" minOccurs="0">
                              <xsd:simpleType>
                                <xsd:restriction base="xsd:decimal">
                                  <xsd:minExclusive value="-999"/>
                                  <xsd:maxExclusive value="999"/>
                                </xsd:restriction>
                              </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="mo" minOccurs="0">
                              <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                  <xsd:maxLength value="12"/>
                                </xsd:restriction>
                              </xsd:simpleType>
                            </xsd:element>
                            <xsd:element name="pkValue1" minOccurs="0">
```

```
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="50"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="pkValue2" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="50"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="pkValue3" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="50"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="pkValue4" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="50"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="pkValue5" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="50"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="logEntryType" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:enumeration value="D1TD"/>
                                      <xsd:enumeration value="F1CR"/>
                                      <xsd:enumeration value="F1ER"/>
                                      <xsd:enumeration value="F1EX"/>
                                      <xsd:enumeration value="F1ST"/>
                                      <xsd:enumeration value="F1SY"/>
                                      <xsd:enumeration value="F1TD"/>
                                      <xsd:enumeration value="F1US"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="logDateTime" nillable="true" type="xsd:dateTime" minOccurs
>
                                <xsd:element name="boStatus" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="12"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="description" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="60"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="user" minOccurs="0">
                                  <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                                      <xsd:maxLength value="8"/>
                                    </xsd:restriction>
                                  </xsd:simpleType>
                                </xsd:element>
                                <xsd:element name="logMessage" minOccurs="0">
```

```
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="4000"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="characteristicType" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="20"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="characteristicValue" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="16"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="adhocValue" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="254"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="fkValue1" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="fkValue2" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="fkValue3" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="fkValue4" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="fkValue5" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="messageCategory" minOccurs="0">
      <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
          <xsd:minExclusive value="-99999"/>
          <xsd:maxExclusive value="99999"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="messageNumber" minOccurs="0">
      <xsd:simpleType>
```

```
                        <xsd:restriction base="xsd:decimal">
                          <xsd:minExclusive value="-99999"/>
                          <xsd:maxExclusive value="99999"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm1" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm2" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm3" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm4" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm5" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm6" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm7" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm8" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                    <xsd:element name="messageParm9" minOccurs="0">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:maxLength value="30"/>
                        </xsd:restriction>
                      </xsd:simpleType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
```

```
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:all>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="deviceEventTypes" minOccurs="0">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="deviceEventTypesList" minOccurs="0" maxOccurs="unbounded">
                    <xsd:complexType>
                      <xsd:sequence>
                        <xsd:element name="deviceEventType" minOccurs="0">
                          <xsd:simpleType>
                            <xsd:restriction base="xsd:string">
                              <xsd:maxLength value="30"/>
                            </xsd:restriction>
                          </xsd:simpleType>
                        </xsd:element>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="boStatusDateTime" nillable="true" type="xsd:dateTime" minOccurs="0"/>
      <xsd:element name="isReprocessPerformed" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="D1NO"/>
            <xsd:enumeration value="D1YS"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="serviceProviderExternalId" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="60"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="syncIMDOtherInfo" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="meterReadSource" type="xsd:string" minOccurs="0"/>
            <xsd:element name="reviewHiLo" type="xsd:string" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="version" minOccurs="0">
        <xsd:simpleType>
          <xsd:restriction base="xsd:decimal">
            <xsd:minExclusive value="-99999"/>
            <xsd:maxExclusive value="99999"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:all>
  </xsd:complexType>
 </xsd:element>

</xsd:all>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Smart Grid Gateway Adapters and Oracle Utilities Cloud Services

This section describes how Oracle Utilities Smart Grid Gateway Adapters are used with Oracle Utilities cloud services such as Oracle Utilities Customer Cloud Service and Oracle Utilities Meter Solution Cloud Service.

Payload Processing

When used with cloud services, Oracle Smart Grid Gateway adapters process usage and device event payloads via batch processing. Files are sent from a head end system to an Oracle Cloud Object Storage location, where they are processed via batch processes.

This process requires set up and configuration of the following:

- **Key Rings** and **Key Pairs**: Records used to link an Oracle Cloud Object Storage location to records in Oracle Utilities cloud services
- **Object Storage Locations**: One or more Oracle Object Storage buckets where payload files are sent from the head end system for processing
- **File Storage Configuration Extendable Lookup Values**: An extendable lookup value that defines the object storage location and allows it to be referenced by batch processes
- **Head End Systems**: A record that defines how specific types of payload processing are handled, such the business object to use to create initial measurements and device events
- **Adapter Extendable Lookups**: Lookup value that define details used by data mapping and processing
- **SGG Payload Processing Configuration Extendable Lookup Values**: An extendable lookup value that defines payload processing details for a specific head end system
- **Payload Processing Batch Controls**: A batch control used for payload process (based on a template batch control provided with the base package)

See the following sections below for more details about each of these.

> **NOTE:** The Itron OpenWay Adapter uses an alternative method for processing usage and device event payloads. See Itron OpenWay Payload Processing for more information.

Creating Key Rings and Pairs

Connection between the Oracle Utilities cloud service and Oracle Object Storage requires an API signature key. See **API Key Management** in the *Oracle Utilities Cloud Services Object Storage Setup Guide* for more information.

API key rings and key pairs are maintained in the **Key Ring** portal. This portal contains the following zones:

- **Key Ring**: Displays basic information about the key ring
- **Key Pairs**: Displays a list of key pairs for the current key ring

Key rings are defined by the following:

- **Key Ring**: A unique code for the key ring
- **Key Ring Class**: Signature (default)
- **Status**: The current status of the key ring. Note: Key pairs can only be generated for Active key rings. Once a key ring has been deactivated, you can no longer create key pairs for that ring.

- **Description**: A name for the key ring (this will be referenced in the File Location extendable lookup)

Once the key ring is created, you need to generate and activate the key pair. Click **Generate Key** to generate a key pair for the key ring.

Key Pairs are defined by the following:

- **Sequence**: The sequence of the key pair (the order in which the key pair was created)
- **Creation Date/Time**: The date/time when the key pair was created
- **Key Status**: The current status of the key pair. Key pairs are inactive when first created.
- **Public Key**: Click **View** to open a dialog box containing the public key.
- **Action**: Click **Activate** to activate an inactive key pair.

Once the key ring and key pair have been created, the key should be pasted into the **User API Key** in **Oracle Identification and Access Management** (IAM) See the **User API Keys** section in the **Security and Access Management** section of the **Managing Object Storage** chapter in the *Oracle Cloud Infrastructure Services* documentation. To copy the public key click **View** in the **Actions** column in the **Key Pairs** zone, and select and copy the text in the **View Public Key** dialog box.

The last step is to activate the key. Click **Activate** in the **Actions** column in the **Key Pairs** zone. A dialog box opens displaying the following message: “Warning(s): Activating a key assumes that you have already registered the public key with the appropriate third parties. Press Cancel to abort.” Note: Be sure to copy the public key to **Oracle Identification and Access Management** before activating it. Click **OK** to activate the key. The **Key Status** column will change to “Active”.

Creating Object Storage Locations

You need to create one or more buckets in Oracle Object Storage where payload files will be sent for processing. Refer to the *Oracle Utilities Cloud Services Object Storage Setup Guide* and the *Oracle Cloud Infrastructure* documentation for more information about creating buckets in Oracle Object Storage.

After creating the buckets in Oracle Object Storage, make a note of the following, as this information is needed in later configuration steps:

- Namespace
- User Oracle Cloud ID (User OCID)
- Tenancy Oracle Cloud ID (Tenancy OCID)
- Compartment Oracle Cloud ID (Compartment OCID)

Suggested Buckets for Payload Processing

This section outlined suggested sets of Object Storage buckets for each of the Smart Grid Gateway adapters.

General Guidelines

As a general rule, each adapter should have separate buckets for each of the following:

- Usage payloads
- Usage archive
- Usage errors
- Device Event payloads
- Device Event archive
- Device Event errors

If an adapter uses payloads in more than one format (such as in CSV and XML formats), you should also consider creating separate buckets for each file format type.

NOTE: If possible, archive buckets should be created in an “Archive” storage tier.

Adapter Development Kit

The table below lists suggested Object Storage buckets for the Adapter Development Kit:

Bucket Name	Storage Tier	Smart Grid Gateway Description
adk-csv	Standard	SGG ADK Adapter - CSV input
adk-csv-arch	Archive	SGG ADK Adapter - CSV archive
adk-csv-error	Standard	SGG ADK Adapter - CSV errors
adk-seeder	Standard	SGG ADK Adapter - IMD Seeder input
adk-seeder-arch	Archive	SGG ADK Adapter - IMD Seeder archive
adk-seeder-error	Standard	SGG ADK Adapter - IMD Seeder errors
adk-xml	Standard	SGG ADK Adapter - XML input
adk-xml-arch	Archive	SGG ADK Adapter - XML archive
adk-xml-error	Standard	SGG ADK Adapter - XML errors

Adapter for Itron OpenWay

The table below lists suggested Object Storage buckets for the Adapter for Itron OpenWay:

Bucket Name	Storage Tier	Smart Grid Gateway Description
itronexception	Standard	SGG Itron OW Adapter - event input
itronexception-arch	Archive	SGG Itron OW Adapter - event archive
itronexception-error	Standard	SGG Itron OW Adapter - event errors
itronxml	Standard	SGG Itron OW Adapter - XML input
itronxml-arch	Archive	SGG Itron OW Adapter - XML archive
itronxml-error	Standard	SGG Itron OW Adapter - XML errors

MV90 Adapter for Itron

The table below lists suggested Object Storage buckets for the MV90 Adapter for Itron

Bucket Name	Storage Tier	Smart Grid Gateway Description
mv90-usage	Standard	SGG MV90 Adapter - usage input
mv90-usage-arch	Archive	SGG MV90 Adapter - usage archive
mv90-usage-error	Standard	SGG MV90 Adapter - usage errors

Adapter for Landis+Gyr

The table below lists suggested Object Storage buckets for the Adapter for Landis+Gyr:

Bucket Name	Storage Tier	Smart Grid Gateway Description
lg-cim-event	Standard	SGG Landis+Gyr Adapter - CIM input
lg-cim-event-arch	Archive	SGG Landis+Gyr Adapter - CIM archive

Bucket Name	Storage Tier	Smart Grid Gateway Description
lg-cim-event-error	Standard	SGG Landis+Gyr Adapter - CIM errors
lg-event	Standard	SGG Landis+Gyr Adapter - event input
lg-event-arch	Archive	SGG Landis+Gyr Adapter - events archive
lg-event-error	Standard	SGG Landis+Gyr Adapter - event errors
lg-usage	Standard	SGG Landis+Gyr Adapter - usage input
lg-usage-arch	Archive	SGG Landis+Gyr Adapter - usage archive
lg-usage-error	Standard	SGG Landis+Gyr Adapter - usage errors

Adapter for Sensus

The table below lists suggested Object Storage buckets for the Adapter for Sensus:

Bucket Name	Storage Tier	Smart Grid Gateway Description
sensus-event	Standard	SGG Sensus Adapter - event input
sensus-event-arch	Archive	SGG Sensus Adapter - events archive
sensus-event-error	Standard	SGG Sensus Adapter - event errors
sensus-usage	Standard	SGG Sensus Adapter - usage input
sensus-usage-arch	Archive	SGG Sensus Adapter - usage archive
sensus-usage-error	Standard	SGG Sensus Adapter - usage errors

Adapter for Silver Spring Networks

The table below lists suggested Object Storage buckets for the Adapter for Silver Spring Networks:

Bucket Name	Storage Tier	Smart Grid Gateway Description
ssn-csv	Standard	SGG SSN Adapter - CSV input
ssn-csv-arch	Archive	SGG SSN Adapter - CSV archive
ssn-csv-error	Standard	SGG SSN Adapter - CSV errors
ssn-ssnxml	Standard	SGG SSN Adapter - XML input
ssn-ssnxml-arch	Archive	SGG SSN Adapter - XML archive
ssn-ssnxml-error	Standard	SGG SSN Adapter - XML errors

Creating File Storage Extendable Lookup Values

Once you have object storage buckets created, the next step is to create an extendable lookup value for each. This record will be referenced by the batch controls that will process the payloads.

File storage configuration information is captured in the File Storage Configuration (F1-FileStorage) extendable lookup.

Values for this extendable lookup are defined by the following:

- **Value**: A unique code for the extendable lookup value. This value will be referenced as a batch control parameter value.
- **Description**: A description of the extendable lookup value

- **Status**: The current status of the value. Select “Active”.
- **File Storage Details**: This section defines details for the object storage location, including:
  - **File Adapter**: The type of file adapter for the location. Select “Oracle Cloud Object Storage”.
  - **User**: The user Oracle Cloud ID (ODIC) for the object storage location
  - **Tenancy**: The tenancy Oracle Cloud ID (ODIC) for the object storage location
  - **Compartment**: The compartment Oracle Cloud ID (ODIC) for the object storage location
  - **Namespace**: The namespace for the object storage location
  - **Key Ring**: The Key Ring you created earlier
  - **Region**: The region of the object storage tenancy for the connection (Values for this field are defined in the F1_REGION_FLG lookup.

Refer to **External File Storage** in the *Oracle Utilities Application Framework Administrative User Guide* and the *Oracle Utilities Cloud Services Object Storage Setup Guide* for more information.

Creating Head End Systems

The next step is to create a record for the head end system. This record defines processing methods for the head end system.

Processing Methods for Payload Processing

The following processing methods should be configured to support payload processing:

- **Device Event Mapping**: defines how head end-specific device events are mapped to standard device event names. The Device Event Seeder business object uses this processing method to determine which type of device event business object to instantiate when receiving device events from a head-end system.
- **Initial Measurement Creation**: defines the business objects used to create initial measurements. The IMD Seeder business object uses this processing method to determine which type of initial measurement business object to instantiate when receiving usage from a head end system.
- **UOM Translation**: defines how head end-specific unit of measure (UOM) codes are mapped to standard UOM codes.

If usage payload include time of use periods or service quantity identifiers, you should also configure **SQI Translation** and **TOU Translation** processing methods.

See Head End Systems and Understanding Processing Methods in the *Administrative User Guide* for more information about creating and configuring a head end system and its processing methods.

Configuring Adapter Extendable Lookups

Smart Grid Gateway adapters use extendable lookups to define data mapping and processing rules specific to each adapter. These include:

- **Device Event Mapping**: used to map head end specific device event codes to standard event names
- **UOM Code to Standard UOM Mapping**: used to map head end specific units of measure (UOM) to standard UOM codes
- **Status Code Mapping**: used to map head end specific interval status codes to standard Condition codes.

Some adapters have additional extendable lookups used to define categories, statuses, and other data.

Creating SGG Payload Processing Extendable Lookup Values

You also have to create an extendable lookup value to define how usage and event payloads will be processed.

Payload processing configuration information is captured in the SGG Payload Processing Configuration (D1-SGGPayloadProcess) extendable lookup.

Values for this extendable lookup are defined by the following:

- **Value**: A unique code for the extendable lookup value. This value will be referenced as a batch control parameter value.
- **Description**: A description of the lookup value
- **Status**: The current status of the value. Set this to “Active”.
- **Processing Details**: This section defines details for payload processing, including
  - **Payload Handler Class Name**: The Java class name for the processing handler to be used. See Payload Handler Classes and Parameters below for more information.
  - **Head-End System**: The head end system for which the configuration will apply. Select from the drop-down list.
  - **Populate Raw**: A checkbox that indicates if the payload data is populated as raw data.
  - **Payload Processing Result Type**: The type of payload to which the configuration applies. Options include:
    - Device Events
    - Initial Measurements
    - Initial Measurements and Device Events
  - **IMD Seeder BO (Interval)**: The seeder business object used to create interval initial measurements. The base package includes the “IMD Seeder” (D1-IMDSeeder) business object for this. Applicable only if the Payload Processing Result Type is “Initial Measurements” or “Initial Measurements and Device Events”.
  - **IMD Seeder BO (Scalar)**: The seeder business object used to create scalar initial measurements. The base package includes the “IMD Seeder” (D1-IMDSeeder) business object for this. Applicable only if the Payload Processing Result Type is “Initial Measurements” or “Initial Measurements and Device Events”.
  - **Device Event Seeder BO**: The seeder business object used to create device events. The base package includes the “Device Event Seeder” (D1-DeviceEventSeeder) business object for this. Applicable only if the Payload Processing Result Type is “Device Events” or “Initial Measurements and Device Events”.
  - **Filter IMD**: A checkbox that indicates if initial measurements should be filtered. Applicable only if the Payload Processing Result Type is “Initial Measurements” or “Initial Measurements and Device Events”. See **Filtering Payloads** below for more information.
  - **Filter Device Events**: A checkbox that indicates if device events should be filtered. Applicable only if the Payload Processing Result Type is “Device Events” or “Initial Measurements and Device Events”. See **Filtering Payloads** below for more information.
- **Dynamic Parameters**: A list of parameter names and values that support additional processing rules, based on the specified Payload Handler Class. See Payload Handler Classes and Parameters below for specific parameters supported by each adapter.
- **User Exit Interceptors**: Defines custom Groovy Library Scripts to be executed via user exit calls when parsing and transforming incoming usage and device data in custom and non-standard formats. User exit interceptors are defined as follows:

Payload Processing User Exit Type	Groovy Library Script	Groovy Library Method
On Get Parser	Parsing script	getParser
On Get Transformer	Transforming script	getTransformer
On Result List	Handler script	getHandler

See Adapter Development Kit Custom Payload Processing for more information about the On Get Parser and On Get Transformer User Exit Interceptors. See Mapping Additional Fields for information about the On Result List User Exit Interceptor.

NOTE: On Get Parser and On Get Transformer User Exit Interceptors are applicable only when the **Payload Handler Class Name** is set to `com.splwg.d1.domain.sgg.dg.processing.PayloadHandlerViaUserExits`.

Payload Handler Classes and Parameters

This section provides valid Payload Handler Class Names and associated dynamic parameters for supported head end systems and payload types.

Adapter Development Kit

As delivered, the Adapter Development Kit supports payload processing of usage and event data exported from an AMI head end system in the "native" initial measurement and device event data formats (the format of the initial measurement and device event seeder business objects). See The Adapter Development Kit Native Format for more information.

Payload Type: ADK CSV

- **Payload Handler Class**: com.splwg.d1.domain.sgg.dg.processing.CSVPayloadHandler

Payload Type: ADK XML

- **Payload Handler Class**: com.splwg.d1.domain.sgg.dg.processing.XMLPayloadHandler

Payload Type: ADK Seeder

- **Payload Handler Class**: com.splwg.d1.domain.sgg.dg.processing.SeederPayloadHandler

Processing Custom and Non-Standard Formats

The Adapter Development Kit can be configured to support payload processing of data from AMI head end systems in custom and non-standard formats. Processing payloads of this type requires a specific **Payload Handler Class** (see below), and creation of Groovy Library Scripts to parse and transform data into the "native" format. See **Adapter Development Kit Custom Payload Processing** and **Payload Processing User Exit Interceptor Scripts** for more information.

- **Payload Handler Class**: com.splwg.d1.domain.sgg.dg.processing.PayloadHandlerViaUserExits

Adapter for Itron OpenWay

Payload Type: Itron OpenWay XML

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d8.processing.ItronXMLPayloadHandler
- **Dynamic Parameters**:

Dynamic Parameter Name	Description and Valid Values
intervalValueDecPlaces	Optional value. Specifies the number of digits after the decimal place for the quantity value. If this parameter is missing, the processing logic defaults the number of digits after the decimal place to 6.
filterRegisterSource	Optional value. Determines if source register data should be filtered. Valid values are 'true' and 'false' (default)

Payload Type: Itron OpenWay Exception

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d8.processing.ExceptionPayloadHandler

MV90 Adapter for Itron

Payload Type: MV90

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d5.processing.MV90PayloadHandler
- **Dynamic Parameters**:

Dynamic Parameter Name	Description and Valid Values
usePhysicalChannel	Optional value. Determines whether the physical channel ID is passed to the IMD seeder to create the measuring component identifier number. If this is set to false, the MV90 LOGCHAN field is used.
fieldForDvcIdN	Optional value. Specifies which field is used as the value for the device ID. Valid values include: • DC_RECID • DC_CUSTID • DC_METERID (default)
MV90ScalarChannelSuffix	Optional value. Holds a suffix value to be added to the measuring component identifier number when a scalarIMD is created for register reads. Default value is "_S"
processMV90ScalarData	Optional value. Determines if register reads are processed. Valid values are 'true' and 'false' (default).
sendStatusAsIs	Optional value. Determines if the status codes will be passed as they came from the head end. If this set to false or not provided, then the default existing behavior when a status code is parsed and status types passed to the IMD seeder in which bits are turned on. Valid values are 'true' and 'false' (default).
MV90DateFormat	Optional value. Holds the date format used to parse provided date-time fields. The code supports the limited list of values for the format. If this parameter is missing or if the value supplied to it does not match the values from this list then the format used will be MDDYYhhmm, the default value. Valid values include: • MMDDYYhhmm (default) • MMYYDDhhmm • DDMMYYhhmm • DDYYMMhhmm • YYMMDDhhmm • YYDDMMhhmm
MV90IntervalValueDecPlaces	Optional value. Specifies the number of digits after the decimal place for the quantity value. If this parameter is missing, the processing logic defaults the number of digits after the decimal place to 6.
MV90StrtmtrAndStopmtrToDecimal	Optional value. Determines if the MV90 STRTMTR (Start Reading Meter) and STOPMTR (Stop Reading Meter) will be converted to a decimal. The processing logic will treat these fields as decimal use the setting to determine how many decimal points are required.

Adapter for Landis+Gyr

Payload Type: L+G Usage

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d3.processing.LGUsagePayloadHandler
- **Dynamic Parameters:**

Dynamic Parameter Name	Description and Valid Values
dateTimeInUTC	Indicates whether the Landis+Gyr system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device. Valid values are 'true' and 'false' (default)

Payload Type: L+G Event

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d3.processing.LGEventPayloadHandler
- **Dynamic Parameters:**

Dynamic Parameter Name	Description and Valid Values
dateTimeInUTC	Indicates whether the Landis+Gyr system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device. Valid values are 'true' and 'false' (default)

Payload Type: L+G CIM Event

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d3.processing.LGCIMEventPayloadHandler
- **Dynamic Parameters:** N/A

Adapter for Sensus

Payload Type: Sensus RNI Usage

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d6.processing.SensusUsagePayloadHandler
- **Dynamic Parameters:**

Dynamic Parameter Name	Description and Valid Values
dateTimeInUTC	Indicates whether the Sensus RNI system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device. Valid values are 'true' and 'false' (default)
useExternalTOU	Indicates whether or not an externally reference TOU period should be used when processing measurement data. Valid values are 'true' and 'false' (default)
sendStatusAsIs	Optional value. Determines if the status codes will be passed as they came from the head end. If this set to false or not provided, then the default existing behavior when a status code is parsed and status types passed to the IMD seeder in which bits are turned on. Valid values are 'true' and 'false' (default)
sendIntervalDateTimes	Optional value. Determines if the "preVEE/msrs/mL/dt" element of the IMD Seeder for interval records exists and contains the last interval

Dynamic Parameter Name	Description and Valid Values
	instead of calculating based on start date + spi * number of intervals. Valid values are ‘true’ and ‘false’ (default)

Payload Type: Sensus RNI Event

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d6.processing.SensusEventPayloadHandler
- **Dynamic Parameters:**

Dynamic Parameter Name	Description and Valid Values
dateTimeInUTC	Indicates whether the Sensus RNI system is sending date/time information in UTC (true) or local time of the device (false). If not provided the default behavior will be local time of the device. Valid values are ‘true’ and ‘false’ (default)

Adapter for Silver Spring Networks

Payload Type: Silver Spring Networks CSV

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d7.processing.SSNCSVPayloadHandler

Payload Type: Silver Spring Networks XML

- **Payload Handler Class**: com.splwg.d1.domain.sgg.d7.processing.SSNXMLPayloadHandler

Common Parameters

The following parameters can be used by all of the supported head end system adapters.

Dynamic Parameter Name	Description and Valid Values
suppressPayloadStatistics	Optional parameter. Determines if the PayloadStatistics business object will not be created while processing payload. If this set to false or not provided, then the default existing behavior when a PayloadStatistics business object is created at the start of the payload processing. Valid values are ‘true’ and ‘false’ (default).
suppressPayloadErrorNotification	Optional parameter. Determines if the PayloadErrorNotification business object will not be created when an error occurs while processing payload. If this set to false or not provided, then the default existing behavior when a PayloadErrorNotification business object is created for every error occurred during the payload processing. If this set to false, the error description will be written into the application log file. Valid values are ‘true’ and ‘false’ (default).
suppressPayloadSummary	Optional parameter. Determines if the PayloadSummary business object will not be created while processing payload. If this set to false or not provided, then the default existing behavior when a PayloadSummary business object is created at the end of the payload processing. Valid values are ‘true’ and ‘false’ (default).
chunkSize	Optional parameter. Specifies the size in bytes of a portion (chunk) of the payload that will be processed in a single batch “thread”. See **Background Processing Overview** for more information about the batch “thread”. This parameter is used when the processing of huge payloads is failing with the “roll back segment too small” error.

Dynamic Parameter Name	Description and Valid Values
	This parameter is supported only for the following payload formats: • L+G Usage • L+G Event • MV90 • Sensus RNI Usage • Sensus RNI Event • Silver Spring Networks CSV • ADK CSV

Filtering Payloads

When **Filter IMD** is selected, only measurements whose <externalUOM> matches one of the values defined in the head end system's UOM Code to Standard UOM Mapping extendable lookup are passed into the system for processing.

When **Filter Device Events** is selected, only device events whose <externalEventName> matches one of the values defined in the head end system's Device Event Mapping extendable lookup are passed into the system for processing.

Upload Statistics Activities

Upload statistics activities are activities that contain statistics related to the processing of usage and device events sent from a head end system.

There are three types of upload statistics activities:

- **Payload Statistics**: Contains statistics related to a specific payload (file) containing one or more initial measurements or device events.
- **Payload Error Notification**: Contains details concerning processing errors encountered in an individual payload (file) containing one or more initial measurements or device events.
- **Payload Summary**: Contains processing summary statistics for an individual payload (file) containing one or more initial measurements or device events. Payload Summary used to update related payload statistics upon the completion of payload processing.

Upload statistics activities are created during processing of payload files as follows:

- When processing begins for a payload, a **Payload Statistics** activity is created to record the process.
- •If an error occurs during processing, a **Payload Error Notification**activity is created.
- •When payload processing is complete, a **Payload Summary**activity is created, which in turn, updates the Payload Statistics activity with details concerning the processing of the payload, including (the start and end time of the processing, the total processing time, the number of initial measurements or device events processed, and the number of initial measurement or device event errors (if any).

Suppressing Upload Statistics Activities

Creation of upload statistics activities can be suppressed through the use of the following dynamic parameters in the SGG Payload Processing Configuration Extendable Lookup:

- suppressPayloadStatistics
- suppressPayloadSummary

- suppressPayloadErrorNotification (**Note**: If this set to false, the error description will be written into the application log file.)

See **Common Parameters** in Payload Handler Classes and Parametersfor more information about these parameters.

Creating Payload Processing Batch Controls

The last step is to create payload processing monitor batch controls based on the Payload Processing Monitor Template (D1-PLPRM) template. You can create separate batch controls for each head end system or for specific payload types, or any combination of both. For example, you might create a separate batch control for Itron OpenWay usage payloads and device event payloads. Creating specific batch controls allows more control when setting up batch process scheduling for payload processing. For instance, you may want to run usage payload processing every hour, but device payload processing every two hours.

To create a payload processing monitor batch control, search for the “Payload Processing Monitor Template” (D1-PLPRM) batch control, click **Duplicate**, enter a unique code for the duplicate batch control and click **OK**.

This new batch control can be used to process payloads received from head end systems.

Payload Processing Batch Control Parameters

Payload processing batch processing is controlled by the following parameters:

Parameter Name	Description	Detailed Description	Required?
includeFiles	Include Files	The pattern for types of files to pick up during polling. The parameter supports the "glob" syntax for using wildcard characters.	True
fileLocation	File Location	The Object Storage bucket to poll files from. This should be the **Value** of the File Location Extendable Lookup value for the appropriate bucket (usage or event).	True
processingConfiguration	Processing Configuration	The **Value** of the SGG Payload Processing Configuration Extendable Lookup value	True
errorLocation	Error Location	The Object Storage bucket where defected portions of payload and related rejection details are stored. This should be the **Value** of the File Location Extendable Lookup value for the appropriate “error” bucket.	True
postProcessingAction	Post Processing Action	Defines what happens to a file after it has been processed: delete (default), archive or rename. If "archive" is selected, an archive bucket should be provided.	False
archiveLocation	Archive Location	The Object Storage bucket where processed files are moved to. Required if the Post Processing Action is "archive". This should be the **Value** of the File Location Extendable Lookup value for the appropriate “archive” bucket.	False
processedFileExtension	Processed File Extension	The text to be appended to the file name after the file has been processed. Default is "PROCESSED".	False
maxRaiseSize	Maximum Raise Size	This parameter defines the maximum number of files that the batch program would submit for processing on each polling cycle. Default is No Maximum.	False

Parameter Name	Description	Detailed Description	Required?
DIST-THD-POOL	Thread Pool	Thread pool name to use if the DEFAULT thread pool is not desired.	False

NOTE: Refer to Creating Object Storage Locations for lists of suggested Object Storage buckets for each adapter.

Mapping Additional Fields

Measurements and device event files received from head end systems can include additional fields containing data to be imported into Oracle Utilities Smart Grid Gateway. These additional fields must be mapped to elements within the measurements and device events processed by Smart Grid Gateway.

This mapping can be performed through use of a custom Groovy script, via the “On Result List” User Exit Interceptor specified in the SGG Payload Processing Configuration Extendable lookup.

Payload Processing Overview

This section provides an overview of the payload processing logic when mapping data in incoming measurements or device events.

Oracle Utilities Meter Data Framework Steps

1. The file upload process program starts reading the file.
2. The upload process creates an instance of the Payload Handler class specified on the SGG Payload Processing Configuration extendable lookup. This class handles the interaction with the User Exit Interceptor script.
3. The upload process invokes the “On Result List” User Exit Interceptor. This executes a “handler” Groovy Library Script that returns an instance of a class that implements the com.splwg.d1.domain.sgg.dg.processing.PayloadProcessingCMHandler interface.
   This "CM payload handler" is initialized with a stream opened for the incoming file and a business object instance of SGG Payload Processing Configuration extendable lookup.
4. The upload process parses data from the incoming document into the “Plain XML” format and transforms the Plain XML data into a set of Initial Measurements (IMDs) or/and Device Events in the “native” XML format and returns each as child nodes added to the Result List parent node.
5. The upload process invokes the CM payload handler, passing the Result List parent node.
6. The CM payload handler transforms data the Result List parent node based on the logic in the Handler script. See **CM Payload Handler Steps**, below.
7. The upload process creates business object instances for all child nodes of the Result List.
8. The process steps (4-9) are repeated until the process returns NULL on step 4 (when there are no further records to process).

CM Payload Handler Steps

1. The CM payload handler reads the given measurement or device event inside the Result List node.
2. The CM payload handler transforms specific data elements inside the measurement or device event (such as replacing a <externalUOM> element with the standard <UOM> element) based on the logic in the handler script, and updates the Result List accordingly.

CM Handler Payload Processing User Exit Interceptor Scripts

Mapping data is performed via a Groovy Library Script that is invoked via a user exit during payload processing. The script used in this process is a referred to as a “handler” script.

Handler Script

The “handler” script is responsible for transforming and mapping data elements within measurements and device events.

This script should include the gethandler method that returns an instance of a class that implements the ccom.splwg.d1.domain.sgg.processing.PayloadProcessingCMHandler interface.

To view details of the PayloadProcessingCMHandler interface in the Application Viewer, select the Java Docs Viewer, select the com.splwg.d1.domain.sgg.dg.processing Java package, and select PayloadProcessingCMHandlerfrom the list of interfaces.

The “handler” script should be defined for the “On Result List” Payload Processing User Exit Type in the User Exit Interceptors section on the SGG Payload Processing Configuration extendable lookup. See Creating SGG Payload Processing Extendable Lookup Values for more information.

Configuration Steps

Configuration of Smart Grid Gateway adapters to support mapping of non-standard data involves the following steps:

1. Create a “handler script” that transforms and maps specific data elements inside the measurement or device event (such as replacing a <externalUOM> element with the standard <UOM> element). See CM Handler Payload Processing User Exit Interceptor Scripts for more information.
2. Create an SGG Payload Processing Configuration extendable lookup value. The **Payload Handler Class Name** should reference one of the usage or event “PayloadHandler” classes listed in the Payload Handler Classes and Parameters section. See Creating SGG Payload Processing Extendable Lookup Values for more information about creating these extendable lookup values..
3. Specify the script for the “On Result List” **Payload Processing User Exit Type** in the **User Exit Interceptors** section on the SGG Payload Processing Configuration extendable lookup. See Creating SGG Payload Processing Extendable Lookup Values for more information.
4. Create other objects and data used with payload processing, including:
   - Key Rings and Pairs (See Creating Key Rings and Pairs)
   - Object Storage Locations (See Creating Object Storage Locations)
   - File Storage Extendable Lookup values (See Creating File Storage Extendable Lookup Values)
   - Head End System (See Creating Head End Systems)
   - Payload Processing Batch Controls (See Creating Payload Processing Batch Controls)

Sample Implementation

The following sample illustrates how additional fields can be mapped into the XML format sent to Smart Grid Gateway. In this case, the unit of measure for incoming measurement data is defined by an <externalUOM>, which must be mapped to the standard <UOM> element.

Sample Script

The below Groovy Script will loop through a list of “D1-IMDSeeder” nodes and check the “externalUom” for a value (GD_KWH in this example), and if equal then the <UOM> element will be created with the <externalUOM> values and the <externalUOM> element will be removed.

Script Type: Groovy Library Script

Step 10: Groovy Imports

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import com.splwg.d1.domain.sgg.processing.PayloadProcessingCMHandler;
import com.splwg.shared.logging.Logger;
import com.splwg.shared.logging.LoggerFactory;
import com.splwg.base.support.scripting.XMLUtils;
```

Step 20: Groovy Library Interface

```
PayloadProcessingCMHandler getHandler()
```

Step 30: Groovy Members

```
public PayloadProcessingCMHandler getHandler() {
  return new CMhandler();
}

public class CMhandler implements PayloadProcessingCMHandler{

  org.w3c.dom.Document onList(org.w3c.dom.Document resultList, org.w3c.dom.Document plainXML) {

  NodeList imdList=resultList.getDocumentElement().getElementsByTagName("D1-IMDSeeder");
  for(int i=0;i<imdList.getLength();i++){
    Element imdEl = (Element) imdList.item(i);

    //check if msrs element exists
    NodeList msrsList = imdEl.getElementsByTagName("msrs");
    if (msrsList.length > 0) {
      //get preVEE
      NodeList preVEENodeList = imdEl.getElementsByTagName("preVEE");
      Element preVEENode = (Element) preVEENodeList.item(0);
      //get externaluom
      NodeList uomList = preVEENode.getElementsByTagName("externalUOM");

      Element uomElm = uomList.length>0 ? uomList.item(0) : null;
      String uom = uomElm!=null ? uomElm.getTextContent() : "";
      //get mcIdN
      NodeList mcIdNList = preVEENode.getElementsByTagName("dvcIdN");
      if ( uom.compareTo("GD_KWH")==0 && mcIdNList.length>0 ) {
        Element externalUOMElm = resultList.createElement("uom");
        externalUOMElm.setTextContent(uom);
        //replace uom with externalUOM
        preVEENode.replaceChild(externalUOMElm, uomElm);
      }
    }
  }
  return resultList;
  }
}
```

Sample File

Below is a sample Landis+Gyr usage file. Note that this file has a"Units" value of "GD_KWH".

```
Record Type~Record Version~Time Stamp~Premise ID~ESIID~Provisioned~Meter ID~Purpose~Commodity~Units~Calculati
```

Plain XML Output

Below is the Plain XML output for the above Landis+Gyr usage file. Note that this file has a <Units> element with a value of "GD_KWH".

```
<MeterReads
xmlns="http://xmlns.oracle.com/LandisGyrUsage">
<MeterRead>
<Origin>IMD_INT_GD_LL_SN100_good1.lg-2020-04-13-07-42-36-092</Origin
>
<ServProvExtRefId>L+G</ServProvExtRefId>
<RecordType>MEPMD01</RecordType>
<RecordVersion>20080519</RecordVersion>
<TimeStamp>110920100120000AM</TimeStamp>
<MeterID>GD_LL_SN100</MeterID>
<Purpose>OK</Purpose>
<Comodity>E</Comodity>
<Units>GD_KWH</Units>
<Interval>000100</Interval>
<Count>24</Count>
<FirstIntervalDateTime>10062010010000AM</FirstIntervalDateTime>
<Data>
<Row v="0.2316"/>
<Row v="0.1416"/>
<Row v="0.2904"/>
<Row v="0.2628"/>
<Row v="0.240"/>
<Row v="0.2796"/>
<Row v="0.2232"/>
<Row v="0.1692"/>
<Row v="0.0552"/>
<Row v="0.24096"/>
<Row v="0.276"/>
<Row v="0.1572"/>
<Row v="0.0612"/>
<Row v="0.3168"/>
<Row v="0.132"/>
<Row v="0.1092"/>
<Row v="0.2532"/>
<Row v="0.2844"/>
<Row v="0.198"/>
<Row v="0.0756"/>
<Row v="0.3084"/>
<Row v="0.15240"/>
<Row v="0.1788"/>
<Row v="0.1716"/>
</Data>
<RawData>MEPMD01~20080519~110920100120000AM~~~~GD_LL_SN100~OK~E~GD_KWH~~000100~24~10062010010000AM~~0.2316~~0.
RawData>
</MeterRead>
</MeterReads>
```

Result List Input

Below is the Result List input sent to the handler script. Note that this file has an <externalUOM> element with a value of "GD_KWH".

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultList>
<D1-IMDSeeder>
<fromDateTime>2010-10-06-00.00.00</fromDateTime>
<preVEE>
<dvcIdN>GD_LL_SN100</dvcIdN>
<externalId>IMD_INT_GD_LL_SN100_good1.lg-2020-04-13-07-42-36-092</externalId>
<externalUOM>GD_KWH</externalUOM>
<stDt>2010-10-06-00.00.00</stDt>
<spi>3600</spi>
<imdType>D1IL</imdType>
```

```
<msrs>
<mL>
<s>1</s><q>0.2316</q>
</mL>
<mL>
<s>2</s><q>0.1416</q>
</mL>
<mL>
<s>3</s><q>0.2904</q>
</mL>
<mL>
<s>4</s><q>0.2628</q>
</mL>
<mL>
<s>5</s><q>0.240</q>
</mL>
<mL>
<s>6</s><q>0.2796</q>
</mL>
<mL>
<s>7</s><q>0.2232</q>
</mL>
<mL>
<s>8</s><q>0.1692</q>
</mL>
<mL>
<s>9</s><q>0.0552</q>
</mL>
<mL>
<s>10</s><q>0.24096</q>
</mL>
<mL>
<s>11</s><q>0.276</q>
</mL>
<mL>
<s>12</s><q>0.1572</q>
</mL>
<mL>
<s>13</s><q>0.0612</q>
</mL>
<mL>
<s>14</s><q>0.3168</q>
</mL>
<mL>
<s>15</s><q>0.132</q>
</mL>
<mL>
<s>16</s><q>0.1092</q>
</mL>
<mL>
<s>17</s><q>0.2532</q>
</mL>
<mL>
<s>18</s><q>0.2844</q>
</mL>
<mL>
<s>19</s><q>0.198</q>
</mL>
<mL>
<s>20</s><q>0.0756</q>
</mL>
<mL>
<s>21</s><q>0.3084</q>
</mL>
<mL>
<s>22</s><q>0.15240</q>
</mL>
<mL>
<s>23</s><q>0.1788</q>
</mL>
<mL>
<s>24</s><q>0.1716</q>
```

```
</mL>
</msrs>
</preVEE>
<serviceProviderExternalId>L+G</serviceProviderExternalId>
</D1-IMDSeeder>
</ResultList>
```

Result List Output

Below is the Result List output sent returned by the handler script. Note that <externalUOM> element had been replaced with a <UOM> element with a value of "GD_KWH".

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultList>
<D1-IMDSeeder>
<fromDateTime>2010-10-06-00.00.00</fromDateTime>
<preVEE>
<dvcIdN>GD_LL_SN100</dvcIdN>
<externalId>IMD_INT_GD_LL_SN100_good1.lg-2020-04-13-07-42-36-092</externalId>
<uom>GD_KWH</uom>
<stDt>2010-10-06-00.00.00</stDt>
<spi>3600</spi>
<imdType>D1IL</imdType>
<msrs>
<mL>
<s>1</s><q>0.2316</q>
</mL>
<mL>
<s>2</s><q>0.1416</q>
</mL>
<mL>
<s>3</s><q>0.2904</q>
</mL>
<mL>
<s>4</s><q>0.2628</q>
</mL>
<mL>
<s>5</s><q>0.240</q>
</mL>
<mL>
<s>6</s><q>0.2796</q>
</mL>
<mL>
<s>7</s><q>0.2232</q>
</mL>
<mL>
<s>8</s><q>0.1692</q>
</mL>
<mL>
<s>9</s><q>0.0552</q>
</mL>
<mL>
<s>10</s><q>0.24096</q>
</mL>
<mL>
<s>11</s><q>0.276</q>
</mL>
<mL>
<s>12</s><q>0.1572</q>
</mL>
<mL>
<s>13</s><q>0.0612</q>
</mL>
<mL>
<s>14</s><q>0.3168</q>
</mL>
<mL>
<s>15</s><q>0.132</q>
</mL>
<mL>
<s>16</s><q>0.1092</q>
```

```
</mL>
<mL>
<s>17</s><q>0.2532</q>
</mL>
<mL>
<s>18</s><q>0.2844</q>
</mL>
<mL>
<s>19</s><q>0.198</q>
</mL>
<mL>
<s>20</s><q>0.0756</q>
</mL>
<mL>
<s>21</s><q>0.3084</q>
</mL>
<mL>
<s>22</s><q>0.15240</q>
</mL>
<mL>
<s>23</s><q>0.1788</q>
</mL>
<mL>
<s>24</s><q>0.1716</q>
</mL>
</msrs>
</preVEE>
<serviceProviderExternalId>L+G</serviceProviderExternalId>
</D1-IMDSeeder>
</ResultList>
```

Itron OpenWay Payload Processing

Payload processing for the Itron OpenWay adapter works differently than it does for other adapters. Instead of using batch processing, Itron payloads are processed as they are received.

Payloads and exceptions sent from the Itron OpenWay head end system are received via inbound web services that invoke business components that process the payloads and exceptions as they are received rather than posting them to object storage locations. See Itron OpenWay Inbound Web Services for more information about the inbound web services used to receive payloads and exceptions from the Itron OpenWay head end system.

Configuration of Itron payload processing requires the same set up and configuration as other adapters, with the exception of payload processing batch controls, which are not used. Some specifics regarding configuration for Itron processing include the following:

- **Object Storage Locations**: Itron OpenWay payload processing uses object storage locations (including Key Rings, Key Pairs and File Storage Configuration extendable lookup values) to capture usage and event payloads or exceptions that encounter errors during processing.
- **Head End Systems and Extendable Lookups**: Itron payload processing uses head end system and extendable lookup configuration in the same manner as other adapters.
- **SGG Payload Processing Configuration Extendable Lookup Values**: These values define payload processing details for the Itron head end system, and are referenced on the Itron Master Configuration (rather than on payload processing batch controls).

See Itron OpenWay Master Configuration for more information about configuring the system to process Itron OpenWay payloads and exceptions.

Adapter Development Kit Custom Payload Processing

This section describes how the Oracle Utilities Smart Grid Gateway Adapter Development Kit can be configured to process custom payload formats, including payloads from non-supported head end systems.

Introduction

As delivered, the Adapter Development Kit supports payload processing of usage and event data exported from an AMI head end system in the “native” initial measurement and device event data formats (the format of the initial measurement and device event seeder business objects).

The Adapter Development Kit can also be configured to support payload processing of data from AMI head end systems in custom and non-standard formats. This involves creation of Groovy Library Scripts to:

- Parse the payload into a “plain XML” format, and
- Transform the plain XML format into the “native” ” initial measurement and device event data formats

These scripts are executed as User Exit Interceptors configured on the SGG Payload Processing Configuration extendable lookup. See Creating SGG Payload Processing Configuration Extendable Lookup Values.

Custom Payload Processing Overview

This section provides an overview of the payload processing logic when converting incoming data into the native format.

Oracle Utilities Meter Data Framework Steps

1. The file upload process program starts reading the file.
2. The upload process creates an instance of the Payload Handler class. This class handles the interaction with the User Exit Interceptor scripts.
3. The upload process invokes the “On Get Parser” User Exit Interceptor. This executes a “parser” Groovy Library Script that returns an instance of a class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadParser` interface.
   This “payload parser” is initialized with a stream opened for the incoming file and a business object instance of SGG Payload Processing Configuration extendable lookup.
4. The upload process invokes the “On Get Transformer” User Exit Interceptor. This executes a “transformer” Groovy Library Script that returns an instance of a class that implements `com.splwg.d1.domain.sgg.dg.processing.PayloadTransformer` interface.
   This “payload transformer” is initialized with a string representing the origin of the payload and business object instance of SGG Payload Processing Configuration extendable lookup.
5. The upload process invokes the payload parser to parse data from the incoming document into the “Plain XML” format. See Payload Parser Steps, below.
6. The upload process invokes the payload transformer, passing the Plain XML data and a Result List parent node.
7. The payload transformer transforms the Plain XML data into a set of Initial Measurements (IMDs) or/and Device Events in the “native” XML format and returns each as child nodes added to the Result List parent node. See Payload Transformer Steps, below.
8. The upload process creates business object instances for all child nodes of the Result List.
9. The process steps (5-9) are repeated until the payload parser returns NULL on step 5 (when there are no further records to process).

Payload Parser Steps

1. The payload parser reads an input stream until it hits the logical end of portion of data (each usage or event record), which could be transformed into Plain XML.
2. The payload parser parses the data and converts into the Plain XML format.

3. The process is performed for each call from the upload process.

Payload Transformer Steps

1. The payload transformer reads the given Plain XML.
2. The payload transformer generates a set of XML nodes representing Initial Measurements (IMDs) or Device Events. Names for nodes are taken from the provided SGG Payload Processing Configuration.
3. The payload transformer adds the generated nodes to the given Result List node.

Payload Processing User Exit Interceptor Scripts

Conversion of data from custom and non-standard formats into the "native" format is performed via a pair of Groovy Library Scripts that are invoked via user exits during payload processing (see Custom Payload Processing Overview). The two scripts used in this process include a "parser" script, and a "transformer" script.

Parser Script

The "parser" script is responsible for parsing the incoming data and converting it into the Plain XML format. This script should include the `getParser` method that returns an instance of a class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadParser` interface.

To view details of the *PayloadParser* interface in the Application Viewer, select the **Java Docs Viewer**, select the `com.splwg.d1.domain.sgg.dg.processing` Java package, and select *PayloadParser* from the list of interfaces.

The "parser" script should be defined for the "On Get Parser" Payload Processing User Exit Type in the **User Exit Interceptors** section on the SGG Payload Processing Configuration extendable lookup. See Creating SGG Payload Processing Extendable Lookup Values for more information.

The base package includes two sample parser scripts that can be used as examples when developing custom parser scripts. See Sample Implementation.

Transformer Script

The "transformer" script is responsible for converting data the Plain XML format into the "native" format. This script should include the `getTransformer` method that returns an instance of a class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadTransformer` interface.

To view details of the *PayloadTransformer* interface in the Application Viewer, select the **Java Docs Viewer**, select the `com.splwg.d1.domain.sgg.dg.processing` Java package, and select *PayloadTransformer* from the list of interfaces.

The "transformer" script should be defined for the "On Get Transformer" Payload Processing User Exit Type in the **User Exit Interceptors** section on the SGG Payload Processing Configuration extendable lookup. See Creating SGG Payload Processing Extendable Lookup Values for more information.

The base package includes two sample transformer scripts that can be used as examples when developing custom transformer scripts. See Sample Implementation.

Configuration Steps

Configuration of the Adapter Development Kit to support conversion of non-standard formats into the "native" format involves the following steps:

1. Create a "parser script" that converts data from the incoming format into Plain XML format. See Payload Processing User Exit Interceptor Scripts for more information.
2. Create a "transformer script" that converts data from the Plain XML format to the "native" format.

See Payload Processing User Exit Interceptor Scripts for more information.

3. Create an SGG Payload Processing Configuration extendable lookup value with a **Payload Handler Class Name** of "com.splwg.d1.domain.sgg.dg.processing.PayloadHandlerViaUserExits".
   See Creating SGG Payload Processing Extendable Lookup Values for more information.
4. Specify the parser script for the "On Get Parser" Payload Processing User Exit Type in the **User Exit Interceptors** section on the SGG Payload Processing Configuration extendable lookup.
   See Creating SGG Payload Processing Extendable Lookup Values for more information.
5. Specify the transformer script for the "On Get Transformer" Payload Processing User Exit Type in the **User Exit Interceptors** section on the SGG Payload Processing Configuration extendable lookup.
   See Creating SGG Payload Processing Extendable Lookup Values for more information.
6. Create other objects and data used with payload processing, including:
   - Key Rings and Pairs (See Creating Key Rings and Pairs)
   - Object Storage Locations (See Creating Object Storage Locations)
   - File Storage Extendable Lookup values (See Creating File Storage Extendable Lookup Values)
   - Head End System (See Creating Head End Systems)
   - Payload Processing Batch Controls (See Creating Payload Processing Batch Controls)

Sample Implementation

The Smart Grid Gateway Adapter Development Kit includes sample parser and transformer scripts that demonstrate how custom payload processing works. These sample scripts include samples for processing data in comma separated values (CSV) format and in XML format.

Comma Separated Values Format

This page describes the comma separated values format sample implementation.

Sample File Format

In this sample, data is stored as a comma separated values file.

Note: Values in this example cannot contain a literal comma character. This restriction is added for the simplicity of file parser implementation. The parsing logic can be re-implemented to avoid this restriction. It will not affect the rest of functionality.

The sample CSV file contains the following record types:

Interval Usage: Used for interval usage measurements

Device Events: Used for device events

Trail record: Used to provide the read date/time and number of interval usage and device event records in the file

Interval Usage Format

#	Field Name	Type	Definition
1	Record Type ID	Constant value "U"	Record type identifier
2	Start Date/Time	Time in the Unix Time format.	Meter read's start date/time
3	End Date/Time	Time in the Unix Time format.	Meter read's end date/time
4	Device Id	Arbitrary Text	Device identifier

#	Field Name	Type	Definition
5	Interval Duration	Numeric Integer	The time interval between readings in seconds (seconds per interval).
6	UOM	Arbitrary Text	Describes the units of the data values
7	Data Entry(s)	Numeric Floating-Point and optional Arbitrary Text separated by ":"	Each record can contain unlimited number of data entry fields. Each data entry is a set of two fields – a reading value and an optional reading status flag. Fields are separated by ":" character. The separator is not present when the reading status flag is not provided.

Device Event Format

#	Field Name	Type	Definition
1	Record Type ID	Constant value "E"	Record type identifier
2	Event Date/Time	Time in the Unix Time format.	The date and time at which the event occurred
3	Device Id	Arbitrary Text	Device identifier
4	Event Name	Arbitrary Text	The primary identification name of the event

Trail Record Format

#	Field Name	Type	Definition
1	Record Type ID	Constant value "T"	Record type identifier
2	Creation Date/Time	Time in the Unix Time format.	Meter read's start date/time
3	Total Records	Numeric Integer	Displays number of records in file

Sample File

```
U,1,86400,DEVICE_DG_0,900,KWH,1:S,2.5,-3.99:R,4:B
E,86400,DEVICE_DG_1,Power Outage
U,2,86400,DEVICE_DG_2,900,KWH,1:S,2,3:R,4:B,5,6,7:A
E,86402,DEVICE_DG_3,Tamper attempt suspected
U,3,86400,DEVICE_DG_4,900,KWH,1:S
T,86401,5
```

Sample Scripts

The sample implementation includes the following sample scripts designed for use with the above CSV format:

DG_SmplPrCSV: This sample script includes one method in the Library Interface – `getParser()`. The method returns an instance of the class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadParse` interface. The class contains the following logic:

1. Parses an incoming payload in the CSV Format.
2. Transforms incoming data to Usage or Event related structures based on incoming data type
3. Returns one by one that structures in the Plain XML format

DG_SmplTrCSV: This sample script includes one method in the Library Interface – `getTransformer()`. The method returns an instance of class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadTransformer` interface. The class contains the following logic:

1. Transforms incoming message in the Plain XML format into the "native" format (the IMD Seeder or Device Event Seeder XML nodes).
2. Adds new XML nodes as child nodes to the given Result List.

Use the **Script** portal to view these scripts in more detail.

Parsing

The parser script performs the following data mapping.

Interval Usage to Plain XML Mapping

The following table shows the mapping between fields in incoming interval data and child elements of Payload/Usage element in the Plain XML format:

Interval Usage Field	Plain XML Element
Record Type ID	RecordType
Start Date/Time	StartDateTime
End Date/Time	EndDateTime
Device Id	DeviceId
Interval Duration	IntervalDuration
UOM	UOM
Data Entry(s)	Intervals/ Interval/ Value [Status]
Record content from incoming file.	RawData

Device Event to Plain XML Mapping

The following table shows the mapping between fields in incoming interval data and child elements of Payload/Event element in the Plain XML format:

Device Event Field	Plain XML Element
Record Type ID	RecordType
Event Date/Time	DateTime
Device Id	DeviceId
Event Name	Name

Device Event Field	Plain XML Element
Record content from incoming file.	RawData

XML Schema of Plain XML Format

The Plain XML consists of the Interval Usage and Device Event related elements at the same time.

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://xmlns.oracle.com/GenericAdapter"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ga="http://xmlns.oracle.com/GenericAdapter">
  <xs:element name="Payload" type="ga:PayloadType"/>
  <xs:complexType name="PayloadType">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Usage" type="ga:UsageType"/>
        <xs:element name="Event" type="ga:EventType"/>
      </xs:choice>
      <xs:element name="RawData"  type="ga:NonEmptyString"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="UsageType">
    <xs:sequence>
      <xs:element name="RecordType" type="ga:RecordTypeUsageType"/>
      <xs:element name="StartDateTime" type="xs:integer"/>
      <xs:element name="EndDateTime" type="xs:integer"/>
      <xs:element name="DeviceId" type="ga:NonEmptyString"/>
      <xs:element name="IntervalDuration" type="xs:integer"/>
      <xs:element name="UOM" type="ga:NonEmptyString"/>
      <xs:element name="Intervals" type="ga:IntervalList"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RecordTypeUsageType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="U"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="IntervalList">
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="Interval" type="ga:IntervalType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="IntervalType">
    <xs:sequence>
      <xs:element name="Value" type="xs:float"/>
      <xs:element name="Status" type="ga:NonEmptyString" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="EventType">
    <xs:sequence>
      <xs:element name="RecordType" type="ga:RecordTypeEventType"/>
      <xs:element name="DateTime" type="xs:integer"/>
      <xs:element name="DeviceId" type="ga:NonEmptyString"/>
      <xs:element name="Name" type="ga:NonEmptyString"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RecordTypeEventType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="E"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NonEmptyString">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

NOTE: Note: A part of incoming file that been used to generate current Plain XML structure will be placed into the Payload/RawData element.

Transformation

The transformer script performs the following mapping.

Plain XML to Seeder XML Mapping

The payload transformer creates the IMD or/and Device Event Seeder XML elements as following:

Usage Mapping: If name of Payload/* node is equal to 'Usage' (interval usage), the script creates an XML element with a name defined in the SGG Payload Processing Configuration field “IMD Seeder BO (Interval)” and the structure according to the following mapping table:

Plain XML Element	IMD Seeder Element	Notes
RecordType	N/A	
DeviceId	dvcIdN	
UOM	externalUOM	
IntervalDuration	spi	
StartDateTime	stDt	Convert to OUAF "dateTime" type
EndDateTime	enDt	Convert to OUAF "dateTime" type
Intervals/		
Interval/		
Value [Status]	msrs	Value
msrs/mL/q		
Status ->		
msrs/mL/sts/stsL/st		
N/A	imdType	Const value 'D1IL'
	externalId	Value stored in setOrigin method.
	serviceProviderExternalId	Value stored in setConfiguration method.
RawData	rawData	If the SGG Payload Processing Configuration has the **Populate RAW** checkbox checked.

Device Event Mapping: If name of Payload/* node is equal to 'Event' (device event data) create an XML element with a name defined in the SGG Payload Processing Configuration field “Device Event Seeder BO” and the structure according to the following mapping table:

Plain XML Element	Device Event Seeder Element	Note
RecordType	N/A	
DateTime	eventDateTime	Convert to OUAF "dateTime" type
DeviceId	externalUOM	
Name	externalEventName	
	externalSourceIdentifier	Value stored in setOrigin method.
	externalSenderId	Value stored in setConfiguration method.

Plain XML Element	Device Event Seeder Element	Note
RawData	rawEventInformation	If the SGG Payload Processing Configuration has the **Populate RAW** checkbox checked.

XML Format

This page describes the XML format sample implementation.

Sample File Format

The file format is described in the Online IMD Upload hint on the **Online IMD and Event Upload** screen. (Select **Menu**, select **Communication**, select **Load IMDs/Events (XML)**).

Sample File

```
<deviceList>
  <device>
    <headEnd>L&G</headEnd>
    <headEndExternalId>L+G</headEndExternalId>
    <deviceId></deviceId>
    <deviceIdentifierNumber>DEV-OUSGG-OSB-DEMO-001</deviceIdentifierNumber>
    <initialMeasurementDataList>
      <initialMeasurementData>
        <preVEE>
          <mcIdN></mcIdN>
          <uom>KWH</uom>
          <stDt>2010-05-19-00.00.00</stDt>
          <enDt>2010-05-19-00.30.00</enDt>
          <spi>900</spi>
          <msrs>
            <mL>
              <s>1</s>
              <q>0.2316</q>
            </mL>
            <mL>
              <s>2</s>
              <q>0.1416</q>
            </mL>
          </msrs>
        </preVEE>
      </initialMeasurementData>
      <initialMeasurementData>
        <initialMeasurementDataId>testValue</initialMeasurementDataId>
          <preVEE>
            <mcIdN></mcIdN>
            <uom>KWH2</uom>
            <stDt>2010-05-19-00.30.00</stDt>
            <enDt>2010-05-19-01.00.00</enDt>
            <spi>900</spi>
            <msrs>
              <mL>
                <s>1</s>
                <q>1.2316</q>
              </mL>
              <mL>
                <s>2</s>
                <q>2.1416</q>
              </mL>
            </msrs>
          </preVEE>
      </initialMeasurementData>
    </initialMeasurementDataList>
  </device>
</deviceList>
```

Sample Scripts

The sample implementation includes the following sample scripts designed for use with the above XML format:

DG_SmplPrXML: This sample script includes one method in the Library Interface – `getParser()`. The method returns an instance of the class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadParser` interface. The class contains the following logic:

1. Parses an incoming payload in XML format
2. Fetches information about current device
3. Breaks an incoming data into separate initial measurement data structures
4. Extends these structures with device information
5. Returns one by one these structures in the Plain XML format

DG_SmplTrXML: This sample script includes one method in the Library Interface – `getTransfprmer()`. The method returns an instance of class that implements the `com.splwg.d1.domain.sgg.dg.processing.PayloadTransformer` interface. The class contains the following logic:

1. Transforms incoming message in the Plain XML format into the IMD Seeder XML nodes
2. Adds new XML nodes as child nodes to the given Result List

Use the **Script** portal to view these scripts in more detail.

Parsing

The XML parser script maps data from the above XML format into the Plain XML format.

XML Schema of Plain XML Format

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
targetNamespace="http://xmlns.oracle.com/GenericAdapter"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ga="http://xmlns.oracle.com/GenericAdapter">
  <xs:element name="Payload" type="ga:PayloadType"/>
  <xs:complexType name="PayloadType">
    <xs:sequence>
      <xs:choice>
        <xs:element name="Usage" type="ga:UsageType"/>
        <xs:element name="Event" type="ga:EventType"/>
      </xs:choice>
      <xs:element name="RawData"  type="ga:NonEmptyString"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="UsageType">
    <xs:sequence>
      <xs:element name="RecordType" type="ga:RecordTypeUsageType"/>
      <xs:element name="StartDateTime" type="xs:integer"/>
      <xs:element name="EndDateTime" type="xs:integer"/>
      <xs:element name="DeviceId" type="ga:NonEmptyString"/>
      <xs:element name="IntervalDuration" type="xs:integer"/>
      <xs:element name="UOM" type="ga:NonEmptyString"/>
      <xs:element name="Intervals" type="ga:IntervalList"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RecordTypeUsageType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="U"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="IntervalList">
```

```
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="Interval" type="ga:IntervalType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="IntervalType">
    <xs:sequence>
      <xs:element name="Value" type="xs:float"/>
      <xs:element name="Status" type="ga:NonEmptyString" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="EventType">
    <xs:sequence>
      <xs:element name="RecordType" type="ga:RecordTypeEventType"/>
      <xs:element name="DateTime" type="xs:integer"/>
      <xs:element name="DeviceId" type="ga:NonEmptyString"/>
      <xs:element name="Name" type="ga:NonEmptyString"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="RecordTypeEventType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="E"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NonEmptyString">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Transformation

The transformer script performs the following mapping.

Plain XML to Seeder XML Mapping

The payload transformer script does the following:

1. Creates an XML element with a name defined in the SGG Payload Processing Configuration field “IMD Seeder BO (Interval)” and copies all elements from the Plain XML as child nodes of new element.
2. Inserts the “externalId” element populated with value stored in `setOrigin` method.
3. Inserts the “serviceProviderExternalId” element populated with value stored in `setConfiguration` method.
4. If the SGG Payload Processing Configuration has the **Populate RAW** checkbox checked, inserts the “rawData” element populated with the textual XML representation of Plain XML element.

Smart Meter Commands

This section provides information about implementing smart meter commands with Oracle Utilities cloud services, including Oracle Utilities Meter Solution Cloud Service.

Device Communication Overview

The basic communication for all business processing is essentially the same. A communication request is sent from the Oracle Utilities application to the head-end system. This request would be for a connect/disconnect, commission/decommission, measurement data, an on-demand read, or another type of request that the head-end system supports. The head-end system receives the message, acts on the request, and returns a reply.

See Communications in the *Business User Guide*, and About Communications in the *Administrative User Guide* for basic information about command activities and communications.

Cloud Service Command Processing

When implementing Smart Grid Gateway adapters with Oracle Utilities cloud services, command processing works slightly differently than it does in on-premises implementations.

The table below provides a brief description of the communication process, and lists example objects used by the Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay. Note that the process outlined below has been simplified for illustrative purposes, and does not reference every step performed in this process.

Step	Process	Example Data
1.	A user initiates a remote connect command for a device. A remote connect activity business object is instantiated for the command.	Activity Business Object: Remote Connect (D1-RemoteConnect)
2.	The remote connect command activity business object creates an outbound communication. The specific type of outbound communication business object created is determined by the head end system (based on the processing role defined in an Enter algorithm of the “Connection Ready” status of the command activity business object’s lifecycle).	Outbound Communication Business Object: Itron - Reconnect Meter (Remote Connect) (D8-ReconnectMeterDR)
3.	The outbound communication creates an outbound message. The specific type of outbound message created is determined by the head end system (based on the processing role defined in an Enter algorithm of the “Awaiting Response” status of the outbound communication business object’s lifecycle).	Outbound Message Type Business Object: Outbound Message for Itron connect command (D8-ItronRemoteConnectOutbndMsg)
4.	The outbound message is sent to the head end system via an External System, Message Sender, and Outbound Message Type. The head end system sends a synchronous response to acknowledge receipt of the request.	External System: Itron OpenWayHead End System Message Sender: Itron – Remote Connect (D8-Reconnect) Outbound Message Type: Itron – Remote Connect (D8-RemoteConnect)
5.	As the status of the command request is updated, the head end system sends periodic messages. These messages are received by an Inbound Web Service which creates instances of an inbound communication business object. The specific type of inbound communication business object created is determined by the Inbound Web Service.	Inbound Communication Business Object: Itron – StatusChanged (D8-StatusChanged)
6.	When the command request has been completed, it sends a status changed inbound communication with a “Completed” status. This transitions the outbound communication business object to its “Completed” state.	Inbound Communication Business Object: Itron – StatusChanged (D8-StatusChanged)
7.	The remote connect command activity business object creates a “results” outbound communication. The specific type of outbound communication business object created is determined by the head end system (based on the processing role defined in an Enter algorithm of the	Outbound Communication Business Object: Itron - Reconnect Meter Results (D8-ReconnectMeterResultDR)

Step	Process	Example Data
	"Connection Ready" status of the command activity business object's lifecycle).	
8.	The outbound communication creates an outbound message. The specific type of outbound message created is determined by the head end system (based on the processing role defined in an Enter algorithm of the "Awaiting Response" status of the outbound communication business object's lifecycle).	Outbound Message Type Business Object: Outbound Message for Itron connect result command (D8-ItronRemoteCntRsltOutMsg)
9.	An Enter Algorithm on the "Evaluate Response" status of the outbound message business object's lifecycle evaluates the response.	Enter Algorithm: Evaluate Get Reconnect Results Response (D8-EVGRCMRST)
10.	The "result" outbound communication creates a completion event to update the status of the device to indicate it has been connected. The specific type of completion event business object created is specified in an Enter algorithm on the "Create Completion Event" Status of the outbound communication business object's lifecycle.	Algorithm: Create Connect Completion Event from Result (D8-CRCNCER) Completion Event Business Object: Connect Device (D1-ConnectDevice)
11.	The outbound communication updates the "Connect/ Disconnect Completion Flag" and the original activity business object. This update is performed by an Enter algorithm on the "Completed" Status of the outbound communication business object's lifecycle.	

In the case of commands that also request usage readings, such as On Demand Read or Scheduled Read, usage readings are sent separately via DataArrived messages separately to the DataSubscriberService Inbound Web Service, where they can be routed to Object storage for payload processing.

Smart Meter Command Flows

This section provides information about the smart meter command flows used by supported Smart Grid Gateway adapters.

Itron OpenWay Command Flows

The table below lists the communication flows used with each Itron OpenWay command, including:

- **Command**: The specific smart meter command
- **Outbound Communication**: The outbound communication business object created by the command. This sends the command request to the head end system.
  - **Result Outbound Communication** The outbound communication business object used to request the result of a command request
- **Inbound Communication**: The inbound communication created by the response from the head end system
- **Completion Event**: The completion event(s) triggered by the command, if any. Completion events are used to update data in the system as a result of a smart meter command. For example, if a command changes the status of a device, the completion event is responsible for making that change.

Command	Outbound Communication	Inbound Communication	Completion Event
Device Commissioning	Itron - Add Meter Definition (Commission) (D8-AddMeterDefinitionsDR)		Device Commissioning Completio Event
Device Decommissioning	Itron - Deregister Meter (Decommission) (D8-DeregisterMeterDR)		Device Decommissioning Completion Event
Remote Connect	Itron - Reconnect Meter (Remote Connect) (D8-ReconnectMeterDR)	Itron - StatusChanged (D8-StatusChanged)	Connect Device Completion Ever
	Result: Itron - Reconnect Meter Result (D8-ReconnectMeterResultDR)		
Remote Disconnect	Itron - Meter Remote Disconnect (D8-DisconnectMeterDR)	Itron - StatusChanged (D8-StatusChanged)	Disconnect Device Completion Event
	Result: Itron - Disconnect Meter Result (D8-DisconnectMeterResultDR)		
On-Demand Read (Scalar)	Itron - Contingency Read (Scalar) (D8-ReadScalarDR)	Itron - StatusChanged (D8-StatusChanged)	Create IMD Completion Event
	Result: Itron - On Demand Read Result (D8-ReadOnDemandReadResultDR)		
On-Demand Read (Interval)	Itron - Contingency Read (Interval) (D8-ReadIntervalDR)	Itron - StatusChanged (D8-StatusChanged)	Create IMD Completion Event
	Result: Itron - On Demand Read Result (D8-ReadOnDemandReadResultDR)		
Scheduled Read (Scalar)	Itron - Interrogate By Group (D8-InterrogateByGroupDR)	Itron - StatusChanged (D8-StatusChanged)	Create IMD Completion Event
	Result: Itron - Interrogate by Group Results (D8- InterrogateByGroupResultDR)		
Scheduled Read (Interval)	Itron - Interrogate By Group (D8-InterrogateByGroupDR)	Itron - StatusChanged (D8-StatusChanged)	Create IMD Completion Event
	Result: Itron - Interrogate by Group Results (D8- InterrogateByGroupResultDR)		
Device Status Check	Itron - Ping By Endpoints (Status Check) (D8-PingByEndpointsDR)	Itron - StatusChanged (D8-StatusChanged)	
	Result: Itron - Ping By Endpoints Result (D8- PingByEndpointsResultDR)		
Multi-Device Status Check	Itron – Ping By Endpoints Multi-Device (D8-PingByEndpointsMultiDvcDR)	Itron – StatusChanged (D8-StatusChanged)	
	Result: Itron – Ping By Endpoints Multi Dvc Result (D8- PingByEndpointsMDResultDR)		
Load Check	Itron – Detect Load Side Voltage (D8-DetectLoadSideVoltgeByMtrDR)	Itron – StatusChanged (D8-StatusChanged)	
	Result: Itron – Detect Load Side Voltage Result (D8- DetLoadSideVoltageMtrRsltDR)		
Read Disconnect State	Itron - Read Disconnect State (D8-ReadDisconStateByMtrDR)	Itron – StatusChanged (D8-StatusChanged)	
	Result: Itron - ReadDisconnectStateByMeter Result (D8-ReadDisconStByMtrResultDR)		

Landis+Gyr Command Flows

The table below lists the communication flows used with each Landis+Gyr command, including:

- **Command**: The specific smart meter command
- **Outbound Communication**: The outbound communication business object created by the command. This sends the command request to the head end system.
- **Inbound Communication**: The inbound communication created by the response from the head end system
- **Completion Event**: The completion event(s) triggered by the command, if any. Completion events are used to update data in the system as a result of a smart meter command. For example, if a command changes the status of a device, the completion event is responsible for making that change.

Command	Outbound Communication	Inbound Communication	Completion Event
Device Commissioning	L+G Add Meter to Inventory(MultiSpeak) (D3-AddMeterToInventryMultiSpDR)		Device Commissioning Completio Event
Device Decommissioning	L+G Meter Remove Notification (MultiSpeak) (D3-MtrRmvNotifMultiSpeakDR)		Device Decommissioning Completion Event
Meter Retire	L+G Meter Retire Notification (D3-MeterRetireNotificationDR)		
Meter Exchange	L+G Meter Exchange Notification (D3-MeterExNotificatnMultiSpDR)		
Remote Connect	L+G Initiate Connect Disconnect DR (D3-InitiateConnectDisconnectDR)	Con/Discon State Change Notif-Direct Route (D3-ConnectDisconStateChgNtfDR)	Connect Device Completion Ever
Remote Disconnect	L+G Initiate Connect Disconnect DR (D3-InitiateConnectDisconnectDR)	Con/Discon State Change Notif-Direct Route (D3-ConnectDisconStateChgNtfDR)	Disconnect Device Completion Event
Demand Reset	L+G Schedule Demand Reset (Multispeak) (D3-ScheduleDemandResetDR)	Reading Changed Notification Direct Route (D3-ReadingChgNotificationDR)	
On-Demand Read (MultiSpeak)	L+G MultiSpeak Initiate Meter Read By Meter Num Direct Route (D3-InitiateMRByMtrNbrDR)	Reading Changed Notification Direct Route (D3-ReadingChgNotificationDR)	Create IMD Completion Event
On-Demand Read (Interval) [CIM]	CIM Meter On Demand Read Direct Route (Interval) (D3-CIMGetLPDataDR)	CIM AsyncReply MeterReads Direct Route (D3-CIMAsyncReplyMeterReadingDR)	Create IMD Completion Event
On-Demand Read (Scalar) [CIM]	CIM Meter On Demand Read Direct Route (Scalar) (D3-CIMMeterOnDemandReadDR)	CIM AsyncReply MeterReads Direct Route (D3-CIMAsyncReplyMeterReadingDR)	Create IMD Completion Event
Device Status Check	CIM Ping Direct Route (D3-CIMPingDR)	CIM AsyncReply MeterReads Direct Route (D3-CIMAsyncReplyMeterReadingDR)	

Sensus Command Flows

The table below lists the communication flows used with each Sensus command, including:

- **Command**: The specific smart meter command
- **Outbound Communication**: The outbound communication business object created by the command. This sends the command request to the head end system.
- **Inbound Communication**: The inbound communication created by the response from the head end system
- **Completion Event**: The completion event(s) triggered by the command, if any. Completion events are used to update data in the system as a result of a smart meter command. For example, if a command changes the status of a device, the completion event is responsible for making that change.

Command	Outbound Communication	Inbound Communication	Completion Event
Device Commissioning	Sensus -Meter Add Notification Direct Route		Device Commissioning Completio Event

Command	Outbound Communication	Inbound Communication	Completion Event
	(D6-MeterAddNotificationDR)		
Device Decommissioning	Sensus -Meter Remove Notification Direct Route (D6-MeterRemoveNotificationDR)		Device Decommissioning Completion Event
Remote Connect	Sensus - Initiate Conn Discon - DR (D6-InitiateConnectDisconnectDR)	Sensus Conn/Discon State Chg Direct Route (D6-ConnectDisconStateChgNtfDR)	Connect Device Completion Ever
Remote Disconnect	Sensus - Initiate Conn Discon - DR (D6-InitiateConnectDisconnectDR)	Sensus Conn/Discon State Chg Direct Route (D6-ConnectDisconStateChgNtfDR)	Disconnect Device Completion Event
On-Demand Read (MultiSpeak)	Sensus - Initiate Meter Read By Meter ID (D6-InitiateMeterByMeterIdDR)	Sensus - Reading Changed Notification (D6-ReadingChgNotificationDR)	Create IMD Completion Event
Device Status Check	Sensus Initiate Outage Detection DR (D6-InitiateOutageDetectionDR)	Sensus Outage Detect Event Notif Direct Route (D6-ODEventNotificationDR)	

Silver Spring Networks Command Flows

The table below lists the communication flows used with each Silver Spring Networks command, including:

- **Command**: The specific smart meter command
- **Outbound Communication**: The outbound communication business object created by the command. This sends the command request to the head end system.
  - **Get Status Outbound Communication**: The outbound communication business object used to request the status of a command request
  - **Result Outbound Communication**: The outbound communication business object used to request the result of a command request
- **Outbound Messages**: The outbound message business objects for one or more outbound messages created by the outbound communication. These should be configured on the Silver Spring Networks external system.
- **Completion Event**: The completion event(s) triggered by the command, if any. Completion events are used to update data in the system as a result of a smart meter command. For example, if a command changes the status of a device, the completion event is responsible for making that change.

Command	Outbound Communication	Outbound Messages	Completion Event
Device Commissioning	SSN - Replace Location (D7-ReplaceLocationDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Replace Location**: SSN - Replace Location (Comm) OB Message (D7-ReplaceLocationOBMsg)	Device Commissioning Completio Event
Device Decommissioning	SSN - Replace Device At Location (Decomm) (D7-ReplDeviceAtLocForDecommDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Replace Device at Location**: SSN - Replace Device At Loc O/B Message: (D7-ReplaceDeviceAtLocOBMsg)	Device Decommissioning Completion Event
Remote Connect	SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Connect Disconnect**:	Connect Device Completion Ever

Command	Outbound Communication	Outbound Messages	Completion Event
		SSN - Connect Disconnect OB Message (D7-ConnectDisconnectOBMsg)	
	Get Status: SSN - Get Status (D7-GetStatus)	SSN - Get Job Status Outbound Message (D7-GetJobStatusOutboundMsg)	
	Result: SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)	SSN - GetConnectDisconnectResult OB Message (D7-GetCntDiscntResultOBMsg)	
Remote Disconnect	SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Connect Disconnect**: SSN - Connect Disconnect OB Message (D7-ConnectDisconnectOBMsg)	Disconnect Device Completion Event
	Get Status: SSN - Get Status (D7-GetStatus)	SSN - Get Job Status Outbound Message (D7-GetJobStatusOutboundMsg)	
	Result: SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)	SSN - GetConnectDisconnectResult OB Message (D7-GetCntDiscntResultOBMsg)	
On-Demand Read (Scalar)	SSN - Add Meter Read Job (Scalar) (D7-AddMeterReadJobScalarDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Add Meter Read Job Scalar**: SSN - Add Meter Read Job Scalar OB Msg Type (D7-AddMeterReadJobScalarOBMsg)	Create IMD Completion Event
	Get Status: SSN - Get Status (D7-GetStatus)	SSN - Get Job Status Outbound Message (D7-GetJobStatusOutboundMsg)	
	Result: SSN - Meter Read Results (Scalar) (D7-MeterReadResultsScalar)	SSN - GetMeterReadResults(Scalar) By JobId (D7-GetSclrMtrRdRstByJobIdOBMsg)	
On-Demand Read (Interval)	SSN - Add Meter Read Job (Interval) (D7-AddMeterReadJobIntervalDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Add Meter Read Job Interval**: SSN - AddMeterReadJob Interval OB Msg Type (D7-AddMeterReadJobIntervalOBMsg)	Create IMD Completion Event
	Get Status: SSN - Get Status (D7-GetStatus)	SSN - Get Job Status Outbound Message (D7-GetJobStatusOutboundMsg)	
	Result: SSN - Meter Read Results (Interval) (D7-MeterReadResultsInterval)	SSN - GetMeterReadResults(Interval) By JobId (D7-GetIntMtrRdRsltByJobIdOBMsg)	
Device Status Check	SSN - Add Ping Job (D7-AddPingJobDR)	**Find Device**: SSN - Find Device Outbound Message (D7-FindDeviceOutboundMsg) **Add Ping Job**: SSN - Add Ping Job OB Message (D7-AddPingJobOBMsg)	
	Get Status: SSN - Get Status (D7-GetStatus)	SSN - Get Job Status Outbound Message (D7-GetJobStatusOutboundMsg)	
	Result: SSN - Ping Results (D7-PingResultsDR)	SSN - Get Ping Results O/B Message (D7-GetPingResultsOBMsg)	

Sending Outbound Communications

Sending outbound communications to a head end system involves configuring the following entities:

- **Outbound Message Types**: Define how outbound messages of a specific type are handled. Outbound message types are based on a specific outbound message business object that defines the data structure of the message. See Defining Outbound Message Types in the *Administrative User Guide* and Creating Outbound Message Types below for more information.
- **Message Senders**: Define details for sending messages to external systems. See Message Senders in the *Administrative User Guide* for more information.
- **External System**: Defines a set of outbound message types that can be sent to the head end system. See External Systems in the *Administrative User Guide* for more information.
- **Head End System**: Defines processing rules for the head end system, such as the specific activities and communications created for smart meter commands. See Head End Systems and Processing Methods in the *Administrative User Guide* and Configuration Processing Methods below for more information.

Creating Outbound Message Types

You must create outbound message types for each command you plan to use in your implementation. The tables below lists the business objects that should be used when creating outbound message types for each Smart Grid Gateway adapter.

Itron OpenWay Outbound Message Types

This table lists the business objects that should be used when creating outbound message types for the Itron OpenWay adapter.

Command	Outbound Message Business Object	"Result" Outbound Message Business Object
Device Commissioning	Outbound Message for Itron Commission (D8-ItronCommissionOutboundMsg)	N/A
Device Decommissioning	Outbound Message for Itron Decommission (D8-ItronDecommissionOutbndMsg)	N/A
Remote Connect	Outbound Message for Itron connect command (D8-ItronRemoteConnectOutbndMsg)	Outbound Message for Itron connect result command (D8-ItronRemoteCntRsltOutMsg)
Remote Disconnect	Outbound Message for Itron disconnect command (D8-ItronRemoteDisconnectOutMsg)	Outbound Message for Itron disconnect result command (D8-ItronRemoteDcntRsltOutMsg)
On-Demand Read (Scalar)	Outbound Message for Itron Contingency Read command (D8-ItronOnDemandReadOutbndMsg)	O/B Msg for Itron Contingency Read Result (D8-ItronOnDemandReadRsltOBMsg)
On-Demand Read (Interval)	Outbound Message for Itron Contingency Read command (D8-ItronOnDemandReadOutbndMsg)	O/B Msg for Itron Contingency Read Result (D8-ItronOnDemandReadRsltOBMsg)
Scheduled Read (Scalar)	Outbound Message for Itron Schedule Read command (D8-ItronInterrogateByGrpOBMsg)	Outbound Message for Itron Schedule Read Result command (D8-ItronInterrogateByGrpRsltOB)
Scheduled Read (Interval)	Outbound Message for Itron Schedule Read command (D8-ItronInterrogateByGrpOBMsg)	Outbound Message for Itron Schedule Read Result command (D8-ItronInterrogateByGrpRsltOB)
Device Status Check	Outbound Message for Itron Device status check (D8-ItronDvcStatusChkOutbndMsg)	O/B Msg for Itron DeviceStatusCheck Result (D8-ItronDvcStatusChkRstlOBMsg)
Multi-Device Status Check	Outbound Message for Itron Multi device status check (D8-ItronDvcStChkMtDvcOutbndMsg)	O/B Message for Itron MultiDeviceStatusCheck Result (D8-ItronDvcStChkMtDvcRsltOBMsg)
Load Check	Outbound Message for Itron Detect Load Side Voltage	Outbound Message for LoadCheckResult Cmd

Command	Outbound Message Business Object	"Result" Outbound Message Business Object
	(D8-ItronDetectLoadSideVolOBMsg)	(D8-ItronDetLSVoltMtrRsltOutMsg)

Landis+Gyr Outbound Message Types

This table lists the business objects that should be used when creating outbound message types for the Landis+Gyr adapter.

Command	Outbound Message Business Object
Device Commissioning	LG - Add Meter To Inventory OB Message D3-AddMeterToInventoryOBMsg
Device Decommissioning	LG - Meter Remove Notification OB Message (D3-MeterRemoveNotifOBMsg)
Meter Retire	LG - Meter Retire Notification OB Message (D3-MeterRetireNotifOBMsg)
Meter Exchange	LG -Meter Exchange Notification OB Message (D3-MeterExchNotificatnOBMsg)
Remote Connect	L+G Initiate Connect Disconnect OB Message (D3-InitConnectDisconnectOBMsg)
Remote Disconnect	L+G Initiate Connect Disconnect OB Message (D3-InitConnectDisconnectOBMsg)
Demand Reset	LG - Schedule Demand Reset OB Message (D3-ScheduleDemandResetOBMsg)
On-Demand Read (MultiSpeak)	L+G MultiSpeak InitiateMeterReadByMeterNumber OB Message (D3-InitMtrReadByMtrNumOBMsg)
On-Demand Read (Interval) [CIM]	CIM AMIRequest GetMeterReads OB Message (D3-CIMAMIRequestGetMtrRdOBMsg)
On-Demand Read (Scalar) [CIM]	CIM AMIRequest GetMeterReads OB Message (D3-CIMAMIRequestGetMtrRdOBMsg)
Device Status Check	CIM AMIRequest GetMeterReads OB Message (D3-CIMAMIRequestGetMtrRdOBMsg)

Sensus Outbound Message Types

This table lists the business objects that should be used when creating outbound message types for the Sensus adapter.

Command	Outbound Message Business Object
Device Commissioning	Sensus - Meter Add Notification OB Message D6-MeterAddNotificationOBMsg
Device Decommissioning	Sensus -Meter Remove Notification OB Msg (D6-MeterRemoveNotificatnOBMsg)
Remote Connect	Sensus Initiate Connect Discon OB Message (D6-InitConnectDisconnectOBMsg)

Command	Outbound Message Business Object
Remote Disconnect	Sensus Initiate Connect Discon OB Message (D6-InitConnectDisconnectOBMsg)
On-Demand Read (MultiSpeak)	Sensus MultiSpeak InitiateMeterReadByMeterNumber OB Message (D6-InitMtrRdMtrIdOBMsg)
Device Status Check	Sensus InitiateOutageDetection OB Message (D6-InitOutageDetectionOBMsg)

Silver Spring Networks Outbound Message Types

This table lists the business objects that should be used when creating outbound message types for the Silver Spring Networks adapter.

Command	Outbound Message Business Object	"Result" Outbound Message Business Object
Device Commissioning	SSN - Replace Location (D7-ReplaceLocationDR)	N/A
Device Decommissioning	SSN - Replace Device At Location (Decomm) (D7-ReplDeviceAtLocForDecommDR)	N/A
Remote Connect	SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)
Remote Disconnect	SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)
On-Demand Read (Scalar)	SSN - Add Meter Read Job Scalar OB Msg Type (D7-AddMeterReadJobScalarOBMsg)	SSN - GetMeterReadResults(Scalar) By JobId (D7-GetSMtrRdRsltJobIdOBMsgTyp)
On-Demand Read (Interval)	SSN - AddMeterReadJob Interval OB Msg Type (D7-AddMtrReadJobIntervalOBMsg)	SSN - GetMeterReadResults(Interval) By JobId (D7-GetIMtrRdRsltJobIdOBMsgTyp)
Device Status Check	SSN - Add Ping Job (D7-AddPingJobDR)	SSN - Ping Results (D7-PingResultsDR)

Creating Message Senders

You must create message senders for each outbound communication that will be sent to the head end system as part of the command communication process.

NOTE:
When creating message senders as part of the command communication process used with cloud service implementations, the **Message Class** MUST be set to **SOAPSNDR** (Sender for real-time HTTP / SOAP messages).

See Message Senders in the *Administrative User Guide* for more information about creating message senders.

Creating an External System

You must create an external system that represents the head end system and defines a set of outbound message types that can be sent to the head end system.

External System Outbound Message Types

You must configure an outbound message type on the external system for each type of message you will send to the head end system. An external system's outbound message types are defined by the following:

- **Outbound Message Type**: The outbound message type created for the message (see Creating Outbound Message Types)
- **Processing Method**: The method by which the message will be sent. This should be set to "Real-time" Smart Grid Gateway outbound communications.
- **Message Sender**: The message sender created for the message (see Creating Message Senders)
- **Message XSL / Response XSL**: XSL files used to transform your message and the response to the message into the appropriate format used by the head end system (Message XSL) and Smart Grid Gateway (Response XSL). Each Smart Grid Gateway adapter includes XSL files for each type of outbound communication.

See External Systems in the *Administrative User Guide* for more information about creating external systems.

Itron OpenWay XSL Files

The table below lists the XSL provided for use with outbound message types based on the outbound message business objects provided with the Itron OpenWay adapter.

Outbound Message Business Object	Message XSL	Response XSL
Outbound Message for Itron Commission (D8-ItronCommissionOutboundMsg)	D8-CommissionRequest.xsl	D8-CommissionResponse.xsl
Outbound Message for Itron Decommission (D8-ItronDecommissionOutbndMsg)	D8-DecommissionRequest.xsl	D8-DecommissionResponse.xsl
Outbound Message for Itron connect command (D8-ItronRemoteConnectOutbndMsg)	D8-ReconnectRequest.xsl	D8-ReconnectResponse.xsl
Outbound Message for Itron connect result command (D8-ItronRemoteCntRsltOutMsg)	D8-GetReconnectRsRequest.xsl	D8-GetReconnectRsResponse.xsl
Outbound Message for Itron disconnect command (D8-ItronRemoteDisconnectOutMsg)	D8-DisconnectRequest.xsl	D8-DisconnectResponse.xsl
Outbound Message for Itron disconnect result command (D8-ItronRemoteDcntRsltOutMsg)	D8-GetDisconnectRsRequest.xsl	D8-GetDisconnectRsResponse.xsl
Outbound Message for Itron Contingency Read command (D8-ItronOnDemandReadOutbndMsg)	D8-ContRdByEndptsRequest.xsl	D8-ContRdByEndptsResponse.xsl
O/B Msg for Itron Contingency Read Result (D8-ItronOnDemandReadRsltOBMsg)	D8-GetContRdByEndptsRsReq.xsl	D8-GetContRdByEndptsRsResp.xsl
Outbound Message for Itron Schedule Read command (D8-ItronInterrogateByGrpOBMsg)	D8-InterrogateByGroupReq.xsl	D8-InterrogateByGroupResp.xsl

Outbound Message Business Object	Message XSL	Response XSL
Outbound Message for Itron Schedule Read Result command (D8-ItronInterrogateByGrpRsltOB)	D8-GetIntByGroupRsRequest.xsl	D8-GetIntByGroupRsResponse.xsl
Outbound Message for Itron Device status check (D8-ItronDvcStatusChkOutbndMsg)	D8-PingByEndpointsRequest.xsl	D8-PingByEndpointsResponse.xsl
O/B Msg for Itron DeviceStatusCheck Result (D8-ItronDvcStatusChkRstlOBMsg)	D8-GetPingByEndptsRsReq.xsl	D8-GetPingByEndptsRsResp.xsl
Outbound Message for Itron Multi device status check (D8-ItronDvcStChkMtDvcOutbndMsg)	D8-PingByEndpointsRequest.xsl	D8-PingByEndpointsResponse.xsl
O/B Message for Itron MultiDeviceStatusCheck Result (D8-ItronDvcStChkMtDvcRsltOBMsg)	D8-GetPingByEndptsRsReq.xsl	D8-GetPingByEndptsRsResp.xsl
Outbound Message for Itron Detect Load Side Voltage (D8-ItronDetectLoadSideVolOBMsg)	D8-DetLSVByMeterRequest.xsl	D8-DetLSVByMeterResponse.xsl
Outbound Message for LoadCheckResult Cmd (D8-ItronDetLSVoltMtrRsltOutMsg)	D8-GetDetLSVByMeterRsReq.xsl	D8-GetDetLSVByMeterRsResp.xsl
Outbound Message for Itron Read Disconnect State D8-ItronReadDisconStateOutMsg	D8-RdDiscStByMtrsRequest.xsl	D8-RdDiscStByMtrsResponse.xsl
Outbound Message for ReadDisconStateMeter Cmd D8-ItronRdDiscStByMtRsltOutMsg	D8-GetRdDiscStByMtrsRsReq.xsl	D8-GetRdDiscStByMtrsRsResp.xsl

Landis+Gyr XSL Files

The table below lists the XSL provided for use with outbound message types based on the outbound message business objects provided with the Landis+Gyr adapter.

Outbound Message Business Object	Message XSL	Response XSL
LG - Add Meter To Inventory OB Message D3-AddMeterToInventoryOBMsg	D3-AddMtrToInvRequest.xsl	D3-AddMtrToInvResponse.xsl
LG - Meter Remove Notification OB Message D3-MeterRemoveNotifOBMsg	D3-MtrRemoveNotfRequest.xsl	D3-MtrRemoveNotfResponse.xsl
LG - Meter Retire Notification OB Message D3-MeterRetireNotifOBMsg	D3-MtrRetireNotfRequest.xsl	D3-MtrRetireNotfResponse.xsl
LG -Meter Exchange Notification OB Message D3-MeterExchNotificatnOBMsg	D3-MtrExcNotifRequest.xsl	D3-MtrExcNotifResponse.xsl
L+G Initiate Connect Disconnect OB Message D3-InitConnectDisconnectOBMsg	D3-InitiateConnDisconReq.xsl	D3-InitiateConnDisconResp.xsl
LG - Schedule Demand Reset OB Message D3-ScheduleDemandResetOBMsg	ScheduleDemandRstReq.xsl	D3-ScheduleDemandRstResp.xsl

Outbound Message Business Object	Message XSL	Response XSL
L+G MultiSpeak InitiateMeterReadByMeterNumber OB Message D3-InitMtrReadByMtrNumOBMsg	D3-InitMtrRdMtrNumRequest.xsl	D3-InitMtrRdMtrNumResponse.xsl
CIM AMIRequest GetMeterReads OB Message D3-CIMAMIRequestGetMtrRdOBMsg	D3-CIMGetMtrReadsRequest.xsl	D3-CIMGetMtrReadsResponse.xsl

Sensus XSL Files

The table below lists the XSL provided for use with outbound message types based on the outbound message business objects provided with the Sensus adapter.

Outbound Message Business Object	Message XSL	Response XSL
Sensus - Meter Add Notification OB Message D6-MeterAddNotificationOBMsg	D6-MtrAddRemoveNotifReq.xsl	D6-MeterAddNotifResponse.xsl
Sensus -Meter Remove Notification OB Msg (D6-MeterRemoveNotificatnOBMsg)	D6-MtrAddRemoveNotifReq.xsl	D6-MeterAddNotifResponse.xsl
Sensus Initiate Connect Discon OB Message (D6-InitConnectDisconnectOBMsg)	D6-InitiateConnDisconReq.xsl	D6-InitiateConnDisconResp.xsl
Sensus Initiate Connect Discon OB Message (D6-InitConnectDisconnectOBMsg)	D6-InitiateConnDisconReq.xsl	D6-InitiateConnDisconResp.xsl
Sensus MultiSpeak InitiateMeterReadByMeterNumber OB Message (D6-InitMtrRdMtrIdOBMsg)	D6-InitMtrRdMtrIdRequest.xsl	D6-InitMtrRdMtrIdResponse.xsl
Sensus InitiateOutageDetection OB Message (D6-InitOutageDetectionOBMsg)	D6-InitOutDetectRequest.xsl	D6-InitOutDetectResponse.xsl

Silver Spring Networks XSL Files

The table below lists the XSL provided for use with outbound message types based on the outbound message business objects provided with the Silver Spring Networks adapter.

Outbound Message Business Object	Message XSL	Response XSL
SSN - Find Device Outbound Message D7-FindDeviceOutboundMsg	D7-FindDeviceRequest.xsl	D7-FindDeviceResponse.xsl
SSN - Get Job Status Outbound Message D7-GetJobStatusOutboundMsgType	D7-GetJobStatusRequest.xsl	D7-GetJobStatusResponse.xsl
SSN - Replace Location (D7-ReplaceLocationDR)	D7-ReplaceLocationRequest.xsl	D7-ReplaceLocationResponse.xsl
SSN - Replace Device At Location (Decomm) (D7-ReplDeviceAtLocForDecommDR)	D7-RepDvcAtLocRequest.xsl	D7-RepDvcAtLocResponse.xsl
SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	D7-AddRemProvJobRequest.xsl	D7-AddRemProvJobResponse.xsl

Outbound Message Business Object	Message XSL	Response XSL
SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)	D7-GetRemProvRsByJobIDReq.xsl	D7-GetRemProvRsByJobIDResp.xsl
SSN - Add Meter Read Job Scalar OB Msg D7-AddMeterReadJobScalarOBMsg	D7-AddMtrRdJobScRequest.xsl	D7-AddMtrRdJobScResponse.xsl
SSN - GetMeterReadResults(Scalar) By JobId D7-GetSclrMtrRdRstByJobIdOBMsg	D7-GetMtrRdRsByJobIDReq.xsl	D7-GetMtrRdRsByJobIDResp.xsl
SSN - AddMeterReadJob Interval OB Msg D7-GetIntMtrRdRsltByJobIdOBMsg	D7-AddMtrRdJobIntRequest.xsl	D7-AddMtrRdJobIntResponse.xsl
SSN - GetMeterReadResults(Interval) By JobId D7-GetIMtrRdRsltJobIdOBMsgTyp	D7-GetMtrRdRsByJobIDReq.xsl	D7-GetMtrRdRsByJobIDResp.xsl
SSN - Add Ping Job (D7-AddPingJobDR)	D7-AddPingJobRequest.xsl	D7-AddPingJobResponse.xsl
SSN - Ping Results (D7-PingResultsDR)	D7-GetMtrRdRsByJobIDReq.xsl	D7-GetMtrRdRsByJobIDResp.xsl

Configuring Processing Methods

In order to send smart meter commands to a head end system, appropriate processing methods must be configured on the head end system record. The tables below list the details appropriate to configuring processing methods for each command.

This table includes the following:

- **Command**: The command you wish to configure. When a command is initiated for a device, the system creates a command activity.
- **Processing Role**: The Processing Role appropriate for the command to be configured.
- **Business Object**: The outbound communication business object to be created by the command activity. Note that most commands have separate business objects for the initial command request and the result request.
- **Outbound Message Business Object** The business object that should be used when creating outbound message types for each command. Note that most commands use separate business objects for the initial outbound message type and the result outbound message type.

Itron OpenWay Processing Methods

This table lists the details appropriate to configuring processing methods for each command supported for the Itron OpenWay head end system.

Command	Processing Role	Business Object	Outbound Message Business Object
Device Commissioning	Device Registration	Itron - Add Meter Definition (D8-AddMeterDefinitionsDR)	Outbound Message for Itron Commission D8-ItronCommissionOutboundMsg
Device Decommissioning	Device Deregistration	Itron - Deregister Meter (D8-DeregisterMeterDR)	Outbound Message for Itron Decommission D8-ItronDecommissionOutbndMsg
Remote Connect	Remote Connect	Itron - Reconnect Meter (Remote Connect) D8-ReconnectMeterDR)	Outbound Message for Itron connect command D8-ItronRemoteConnectOutbndMsg

Command	Processing Role	Business Object	Outbound Message Business Object
	Get Remote Connect Result	Itron - Reconnect Meter Result (D8-ReconnectMeterResult)	Outbound Message for Itron connect result command (D8-ItronRemoteCntRsltOutMsg)
Remote Disconnect	Remote Disconnect	Itron - Disconnect Meter (D8-DisconnectMeterDR)	Outbound Message for Itron disconnect command D8-ItronRemoteDisconnectOutMsg
	Get Remote Disconnect Result	Itron - Disconnect Meter Result (D8-DisconnectMeterResult)	Outbound Message for Itron disconnect result command (D8-ItronRemoteDcntRsltOutMsg)
On-Demand Read (Scalar)	On-Demand Read (Scalar)	Itron - Contingency Read (Scalar) (D8-ReadScalarDR)	Outbound Message for Itron Contingency Read command D8-ItronOnDemandReadOutbndMsg
	Get On-Demand Read Result	Itron - On Demand Read Result (D8-ReadOnDemandReadResultDR)	O/B Msg for Itron Contingency Read Result (D8-ItronOnDemandReadRsltOBMsg)
On-Demand Read (Interval)	On-Demand Read (Interval)	Itron - Contingency Read (Interval) (D8-ReadIntervalDR)	Outbound Message for Itron Contingency Read command (D8-ItronOnDemandReadOutbndMsg)
	Get On-Demand Read Result	Itron - On Demand Read Result (D8-ReadOnDemandReadResultDR)	O/B Msg for Itron Contingency Read Result (D8-ItronOnDemandReadRsltOBMsg)
Scheduled Read (Scalar)	Send Schedule Read Request	Itron - Interrogate by Group (Scalar) (D8-InterrogateByGroupDR)	Outbound Message for Itron Schedule Read command (D8-ItronInterrogateByGrpOBMsg)
	Get Schedule Read Result	Itron - Interrogate by Group Results (D8- InterrogateByGroupResultDR)	Outbound Message for Itron Schedule Read Result command (D8-ItronInterrogateByGrpRsltOB)
Scheduled Read (Interval)	Send Schedule Read Request	Itron - Interrogate by Group (Interval) (D8-InterrogateByGroupDR)	Outbound Message for Itron Schedule Read command (D8-ItronInterrogateByGrpOBMsg)
	Get Schedule Read Result	Itron - Interrogate by Group Results (D8- InterrogateByGroupResultDR)	Outbound Message for Itron Schedule Read Result command (D8-ItronInterrogateByGrpRsltOB)
Device Status Check	Device Status Check	Itron - Ping by Endpoints (Status Check) (D8-PingByEndpointsDR)	Outbound Message for Itron Device status check (D8-ItronDvcStatusChkOutbndMsg)
	Get Device Status Check Result	Itron - Ping By Endpoints Result (D8- PingByEndpointsResultDR)	O/B Msg for Itron DeviceStatusCheck Result (D8-ItronDvcStatusChkRstlOBMsg)
Multi-Device Status Check	Multi-Device Status Check	Itron - Ping By Endpoints Multi-Device (D8-PingByEndpointsMultiDvcDR)	Outbound Message for Itron Multi device status check (D8-ItronDvcStChkMtDvcOutbndMsg)
	Get Multi-Device Status Check Result	Itron – Ping By Endpoints Multi Dvc Result (D8- PingByEndpointsMDResultDR)	O/B Message for Itron MultiDeviceStatusCheck Result (D8-ItronDvcStChkMtDvcRsltOBMsg)
Load Check	Load Check	Itron - Detect Load Side Voltage (D8-DetectLoadSideVoltgeByMtrDR)	Outbound Message for Itron Detect Load Side Voltage (D8-ItronDetectLoadSideVolOBMsg)
	Get Load Check Results	Itron – Detect Load Side Voltage Result (D8- DetLoadSideVoltageMtrRsltDR)	Outbound Message for LoadCheckResult Cmd (D8-ItronDetLSVoltMtrRsltOutMsg)

Landis+Gyr Processing Methods

This table lists the details appropriate to configuring processing methods for each command supported for the Landis+Gyr head end system.

Command	Processing Role	Business Object	Outbound Message Business Object
Device Commissioning	Device Registration	L+G Add Meter to Inventory(MultiSpeak) (D3-AddMeterToInventryMultiSpDR)	LG - Add Meter To Inventory OB Message D3-AddMeterToInventoryOBMsg
Device Decommissioning	Device Removal	L+G Meter Remove Notification (MultiSpeak) (D3-MtrRmvNotifMultiSpeakDR)	LG - Meter Remove Notification OB Message (D3-MeterRemoveNotifOBMsg)
Meter Retire	Device Deregistration	L+G Meter Retire Notification (D3-MeterRetireNotificationDR)	LG - Meter Retire Notification OB Message (D3-MeterRetireNotifOBMsg)
Meter Exchange	Device Installation	L+G Meter Exchange Notification (D3-MeterExNotificatnMultiSpDR)	LG -Meter Exchange Notification OB Message (D3-MeterExchNotificatnOBMsg)
Remote Connect	Remote Connect	L+G Initiate Connect Disconnect DR (D3-InitiateConnectDisconnectDR)	L+G Initiate Connect Disconnect OB Message (D3-InitConnectDisconnectOBMsg)
Remote Disconnect	Remote Disconnect	L+G Initiate Connect Disconnect DR (D3-InitiateConnectDisconnectDR)	L+G Initiate Connect Disconnect OB Message (D3-InitConnectDisconnectOBMsg)
Demand Reset	Demand Reset	L+G Schedule Demand Reset (Multispeak) (D3-ScheduleDemandResetDR)	LG - Schedule Demand Reset OB Message (D3-ScheduleDemandResetOBMsg)
On-Demand Read (MultiSpeak)	On-Demand Read (Scalar)	L+G MultiSpeak Initiate Meter Read By Meter Num Direct Route (D3-InitiateMRByMtrNbrDR)	L+G MultiSpeak InitiateMeterReadByMeterNumber OB Message (D3-InitMtrReadByMtrNumOBMsg)
On-Demand Read (Interval) [CIM]	On-Demand Read (Interval)	CIM Meter On Demand Read Direct Route (Interval) (D3-CIMGetLPDataDR)	CIM AMIRequest GetMeterReads OB Message (D3-CIMAMIRequestGetMtrRdOBMsg)
On-Demand Read (Scalar) [CIM]	On-Demand Read (Scalar)	CIM Meter On Demand Read Direct Route (Scalar) (D3-CIMMeterOnDemandReadDR)	CIM AMIRequest GetMeterReads OB Message (D3-CIMAMIRequestGetMtrRdOBMsg)
Device Status Check	Device Status Check	CIM Ping Direct Route (D3-CIMPingDR)	CIM AMIRequest GetMeterReads OB Message (D3-CIMAMIRequestGetMtrRdOBMsg)

Sensus Processing Methods

This table lists the details appropriate to configuring processing methods for each command supported for the Sensus head end system.

Command	Processing Role	Business Object	Outbound Message Business Object
Device Commissioning	Device Registration	Sensus -Meter Add Notification Direct Route (D6-MeterAddNotificationDR)	Sensus - Meter Add Notification OB Message D6-MeterAddNotificationOBMsg
Device Decommissioning	Device Removal	Sensus -Meter Remove Notification Direct Route (D6-MeterRemoveNotificationDR)	Sensus -Meter Remove Notification OB Msg (D6-MeterRemoveNotificatnOBMsg)
Remote Connect	Remote Connect	Sensus - Initiate Conn Discon - DR (D6-InitiateConnectDisconnectDR)	Sensus Initiate Connect Discon OB Message (D6-InitConnectDisconnectOBMsg)

Command	Processing Role	Business Object	Outbound Message Business Object
Remote Disconnect	Remote Disconnect	Sensus - Initiate Conn Discon - DR (D6-InitiateConnectDisconnectDR)	Sensus Initiate Connect Discon OB Message (D6-InitConnectDisconnectOBMsg)
On-Demand Read (MultiSpeak)	On-Demand Read (Scalar)	Sensus - Initiate Meter Read By Meter ID (D6-InitiateMeterByMeterIdDR)	Sensus MultiSpeak InitiateMeterReadByMeterNumber OB Message (D6-InitMtrRdMtrIdOBMsg)
Device Status Check	Device Status Check	Sensus Initiate Outage Detection DR (D6-InitiateOutageDetectionDR)	Sensus InitiateOutageDetection OB Message (D6-InitOutageDetectionOBMsg)

Silver Spring Networks Processing Methods

This table lists the details appropriate to configuring processing methods for each command supported for the Silver Spring Networks head end system.

Command	Processing Role	Business Object	Outbound Message Business Object
Device Commissioning	Device Registration	SSN - Replace Location (D7-ReplaceLocationDR)	SSN - Replace Location (Comm) OB Message (D7-ReplaceLocationOBMsg)
Device Decommissioning	Device Removal	SSN - Replace Device At Location (Decomm) (D7-ReplDeviceAtLocForDecommDR)	SSN - Replace Device At Loc O/B Message: (D7-ReplaceDeviceAtLocOBMsg)
Remote Connect	Remote Connect	SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	SSN - Connect Disconnect OB Message (D7-ConnectDisconnectOBMsg)
	Get Remote Connect Result	SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)	SSN - GetConnectDisconnectResult OB Message (D7-GetCntDiscntResultOBMsg)
Remote Disconnect	Remote Disconnect	SSN - Connect or Disconnect (D7-ConnectDisconnectDR)	SSN - Connect Disconnect OB Message (D7-ConnectDisconnectOBMsg)
	Get Remote Disconnect Result	SSN - Connect Disconnect Results (D7-ConnectDisconnectResultDR)	SSN - GetConnectDisconnectResult OB Message (D7-GetCntDiscntResultOBMsg)
On-Demand Read (Scalar)	On-Demand Read (Scalar)	SSN - Add Meter Read Job (Scalar) (D7-AddMeterReadJobScalarDR)	SSN - Add Meter Read Job Scalar OB Msg Type D7-AddMeterReadJobScalarOBMsg
	Get On-Demand Read Result (Scalar)	SSN - Meter Read Results (Scalar) (D7-MeterReadResultsScalar)	SSN - GetMeterReadResults(Scalar) By JobId (D7-GetSMtrRdRsltJobIdOBMsgTyp)
On-Demand Read (Interval)	On-Demand Read (Interval)	SSN - Add Meter Read Job (Interval) (D7-AddMeterReadJobIntervalDR)	SSN - AddMeterReadJob Interval OB Msg Type (D7-AddMtrReadJobIntervalOBMsg)
	Get On-Demand Read Result (Interval)	SSN - Meter Read Results (Interval) (D7-MeterReadResultsInterval)	SSN - GetMeterReadResults(Interval) By JobId (D7-GetIMtrRdRsltJobIdOBMsgTyp)
Device Status Check	Device Status Check	SSN - Add Ping Job (D7-AddPingJobDR)	SSN - Add Ping Job OB Message (D7-AddPingJobOBMsg)
	Get Device Status Check Result	SSN - Ping Results (D7-PingResultsDR)	SSN - Get Ping Results O/B Message (D7-GetPingResultsOBMsg)

Receiving Inbound Communications

Receiving inbound communications from a head end system involves configuring the following:

- **Inbound Web Services**: These services are used to receive inbound communications from external systems. See Inbound Web Services in the *Administrative User Guide* for more information.
- **Master Configurations**: Defines configuration settings for individual adapters.

Inbound Web Services

Inbound web services are used to receive inbound communications from external systems. The tables below list the inbound web services provided with each Smart Grid Gateway adapter.

Itron OpenWay Inbound Web Services

The table below lists the inbound web service provided with the Itron OpenWay adapter.

Name	Description
D8-RequestStatusChangedService	Itron RequestStatusChangedService: Use to receive “status changed” messages from the Itron OpenWay head end system. This service creates instances of the “Itron – StatusChanged” (D8-StatusChanged) inbound communication business object for each message.
D8-DataSubscriberService	Itron DataSubscriberService: Used to receive usage and device event payloads from the Itron OpenWay head end system. This service invokes the “Itron – DataArrived Request” (D8_DATAARR) service script, which invokes the ItronDataArrivedStreamProcessor business component to process the usage or device event payload based on the configuration of the Itron Master Configuration, the SGG Payload Processing Configuration extendable lookup, and the File Storage extendable lookup.
D8-ExceptionSubscriberService	Itron ExceptionSubscriberService: Used to receive exceptions from the Itron OpenWay head end system. This service invokes the “Itron – ExceptionsArrived Request” (D8_EXCPARR) service script, which invokes the ItronExceptionsArrivedStreamProcessor business component to process the exception based on the configuration of the Itron Master Configuration, the SGG Payload Processing Configuration extendable lookup, and the File Storage extendable lookup.

Landis+Gyr Inbound Web Services

The table below lists the inbound web services provided with the Landis+Gyr adapter.

Name	Schema Name	Request / Response XSL
CD State Changed Notification (Callback) (D3-CB_CD)	D3-ConnectDisconStateChgNtfDR	D3-ConDiscnStateChgNtfReq.xsl D3-ConDiscnStateChgNtfResp.xsl
Reading Changed Notification (Callback) (D3-CB_MR)	D3-ReadingChgNotificationDR	D3-ReadingChgNotifRequest.xsl D3-ReadingChgNotifResponse.xsl
CIM Service Callback	D3-CIMAsyncReplyMeterReadingDR	D3-CIMMeterReadingRequest.xsl

Name	Schema Name	Request / Response XSL
(D3-CIMCallback)		D3-CIMMeterReadingResponse.xsl
Schedule Demand Response (Multispeak)	D3-ReadingChgNotificationDR	D3-ReadingChgNotifRequest.xsl
(D3-ScheduleDemandResetResponse)		D3-ReadingChgNotifResponse.xsl

Sensus Inbound Web Services

The table below lists the inbound web services provided with the Sensus adapter.

Name	Schema Name	Request / Response XSL
Reading Changed Notification (Callback)	D6-ReadingChgNotificationDR	D6-ReadingChgNotifRequest.xsl
(D6-CB_MR)		D6-ReadingChgNotifResponse.xsl
Sensus MultiSpeak CB_Server Interface	D6-ConnectDisconStateChgNtfDR	D6-ConDiscnStateChgNtfReq.xsl
(D6-CB_Server)		D6-ConDiscnStateChgNtfResp.xsl
	D6-ReadingChgNotificationDR	D6-ReadingChgNotifRequest.xsl
		D6-ReadingChgNotifResponse.xsl
Sensus MultiSpeak OA_Server Interface	D6-ODEventNotificationDR	D6-ODEventNotifRequest.xsl
(D6-OA_Server)		D6-ODEventNotifResponse.xsl
Unsolicited Event Response	D6-UnsolicitedEvtNotification	
D6-UnsolicitedEventNotification		

Adapter Development Kit - Creating Custom Commands

This section describes how the Oracle Utilities Smart Grid Gateway Adapter Development Kit can be configured to support custom smart meter commands, including those from non-supported head end systems.

This includes:

- An overview of the data and objects used by smart meter commands
- A description and example of synchronous commands
- A description and example of asynchronous commands

Overview

This section provides an overview of how smart meter commands work and the types of objects used in executing commands when implementing Smart Grid Gateway adapters with Oracle Utilities cloud services.

As noted inDevice Communication Overview, command processing works slightly differently than it does in on-premises implementations. That section provides an introduction to command processing in Smart Grid Gateway adapters, and the **Cloud Service Command Processing** section provides a step-by-step description of the various data and objects created as part of command processing.

The remainder of this section provides additional details regarding the three primary types of data and objects used with smart meter command processing:

- Command Activities
- Outbound Communications

- Inbound Communications

Command Activities

Smart meter commands are managed by activities. Activities orchestrate one or more communications and execute any completion events associated with the command.

Key Algorithms

This section outlines some of the important algorithms executed as part of the command activity's lifecycle.

Some algorithms are directly shared by multiple business objects and multiple lifecycle states. In other cases, unique algorithms will perform the same role across business objects. The following section refers to the algorithms generically.

Lifecycle State	Algorithms
Validate	The Validate state is an opportunity to check whether the command should execute. • Check Device Command Eligibility: Checks whether the command request is eligible for the device by calling the Determine Eligible Commands Business Service (D1-DetermineEligibleCommands). The command request is considered eligible if it exists in the eligible commands list returned by the business service. • Validate Head End's Capability to Perform Activity: This algorithm checks to ensure that the head end system has the capability to support the command requested. It accomplishes this by invoking the Determine Service Providers and Methods Business Service. • Check for Existing Active Request: This algorithm looks for non-Final Activities with the same Device and of the same type. This prevents duplicate requests from being sent to the head end system.
Validation Error	If the Activity fails validation, it transitions to the Validation Error state. From this state the activity can be retried and create To Do entriesto notify users of the error. • Retry BO In Error: Reprocesses the BO by transitioning it to its original state. • Create To Do Entry for BO in Error: This algorithm attempts to create a To Do entry using the To Do type and role specified by the algorithm parameters. • Generic To Do Completion for BOs: This algorithm completes all To Do entries with Drill Keys = Current Business Object's prime keys except for To Do Types with Characteristic Type and Value as specified by algorithm parameters. This prevents the existence of multiple To Do entries for the same command and device.
Waiting For Effective Date	Prior to sending the request to the head end system, the activity enters this state. • Wait Time Out - Transition to Rejection: This algorithm rejects the business object if the expiration date time has been reached. • Wait for Effective Date: When the required date is reached, this algorithm transitions the business object to the next configured state.

Lifecycle State	Algorithms
	• Send Response to External Requester: This algorithm sends an acknowledging received response outbound message to the requester if the business object is configured to do so.
Connection Ready	The Connection Ready state shows that the activity can be executed and creates the proper outbound communication business object. This state can be re-entered when there are multiple outbound communications that are required by the head end system's API. • Create Outbound Communication: This algorithm creates an outbound communication instance if the value of the completion flag specified by an algorithm parameter is false. • Create Results Outbound based on Completion Flag: This algorithm triggers a secondary communication to the head end system based on the value of an algorithm parameter. Typically, this indicates the successful completion of an initial outbound communication.
Communication In Progress	This lifecycle state indicates that an outbound communication is in-progress. • Wait Time Out - Transition to Exception: If the business object has been in the In Progress state too long, this algorithm transitions the business object to an exception state. • Check Children Communications: If all of the outbound communication business object instances related to this activity have completed, the algorithm transitions the business object to the next state.
Communication Error	An error-handling lifecycle state to which the business object transitions when the communication encounters a problem. The algorithms are similar to those in the Validation Error state. • Wait Time Out - Transition to Rejection: If the business object has been in the In Progress state too long, this algorithm transitions the business object to a rejection state. • Retry BO In Error: This algorithm reprocesses the business object by transitioning it to its original state. • Create To Do Entry for BO in Error: This algorithm attempts to create a To Do entry using the To Do type and role specified by the algorithm parameters. • Generic To Do Completion for BOs: This algorithm completes all To Do entries with Drill Keys = Current Business Object's prime keys except for To Do Types with Characteristic Type and Value as specified by algorithm parameters. This prevents the existence of multiple To Do entries for the same command and device
Retry	The lifecycle state responsible for discarding existing outbound communications and transitioning the activity back to Connection Ready.
Execute Completion Events	This lifecycle state collates the information acquired by the communications and creates some result. For example, in a DeviceStatusCheck, the overall results are set.

Lifecycle State	Algorithms
	• Execute Completion Events: Fires the completion events associated with the activity.
Completion Event Error	This is another error-handling state similar to the Validation Error and the Communication Error states. The algorithms are the same as in those states.
Completed	This is the final state representing a successful activity. Finishing tasks are performed and external requesters are notified.
Discarded	Similarly to the Completed state, the Discarded state finishes the activity and indicates an overall failure. External requesters are notified of the failure and other finishing tasks are performed.

Outbound Communications

Outbound Communications represent the business logic of sending a request and receiving a reply. Outbound communication business objects are based on the D1-COMMOUT maintenance object. The outbound communication is not the actual message, but creates the outbound message. Outbound communication business objects used with cloud services are suffixed with “-DR” for “direct route” to indicate the request go straight to the head end system.

Schema

This section provides details concerning outbound communication business object schemas.

Outbound communication schemas typically have a “sendDetail” element corresponding to the data being sent to the head end system, or more precisely, the XSL handler for the outbound message. The synchronous response is transformed by the XSL handler and placed into the “responseDetail” element.

Key Algorithms

This section outlines some of the important algorithms executed as part of the outbound communication’s lifecycle.

Some algorithms are directly shared by multiple business objects and multiple lifecycle states. In other cases, unique algorithms will perform the same role across business objects. The following section refers to the algorithms generically.

Lifecycle State	Algorithms
Validate	Similar to the sctivity, the Validate state houses algorithms that will execute prior to sending the message. • Validate Communication Type: This algorithm ensures the communicationType field is populated on the message.
Validation Error	If the Validate state encounters an error, the business object transitions to this state. The algorithms are similar to those in the activity business object.
Awaiting Response	The Awaiting Response state sends an outbound message and awaits the response. • Populate Send Detail: Each outbound communication has a specific algorithm to populate all the schema elements that are required to create the outbound message. • Create Outbound Message: This algorithm creates a specific outbound message based on the Processing Role. The response is stored in the responseDetails element.

Lifecycle State	Algorithms
	• Timeout: This algorithm detects whether a communication has been waiting for an asynchronous response too long.
Retry	The lifecycle state responsible for transitioning the outbound communication back to Awaiting Response.
Response Error	The business object enters the Response Error state when an error has occurred in sending the request. It contains algorithms for creating To Do entries and for retrying similar to the activity's Communication Error state.
Completed	This is the final state representing a successful outbound communication. Finishing tasks such as setting flags for use by subsequent communications and notifying the parent activity of completion are performed.
Discarded	Includes an algorithm that transfers the parent activity to Failed.

Inbound Communications

Inbound communication business objects are based on the D1-COMMIN maintenance objects. Inbound Communication objects have a much less defined schema since they are driven by the needs of the head end system's API and those of the business process. As with outbound communications, the schema does not represent the exact message, but is the post-processed XSL result.

Key Algorithms

This section outlines some of the important algorithms executed as part of the inbound communication's lifecycle.

Some algorithms are directly shared by multiple business objects and multiple lifecycle states. In other cases, unique algorithms will perform the same role across business objects. The following section refers to the algorithms generically.

Lifecycle State	Algorithms
Validate	Similar to the activity, the Validate state houses algorithms that will execute as the message is created. • Validate Communication Type: This algorithm ensures the communicationType field is populated on the message. • Find Parent Outbound Communication for Async Inbound: This algorithm is responsible for linking the incoming message with the original outbound communication using, for example, a transaction ID. This algorithm is what causes an inbound communication to be linked with a larger process.
Validation Error	If the Validate state encounters an error, the business object transitions to this state. The algorithms are similar to those in the activity business object.
Evaluate Response	The Evaluate Response state processes the inbound communication in the context of an "asynchronous response." The algorithms should perform any required update tasks such as setting flags on the parent outbound communication.
Completed	This successful end state for inbound communications should transition the parent outbound communication to the next state. This typically

Lifecycle State	Algorithms
	completes the outbound communication and may trigger the activity to create a subsequent outbound communication.
Discarded	The parent outbound communication should transition to a failure state. This will also place the activity into the Communication Error state where it will await corrections or a Retry attempt.

Synchronous Commands

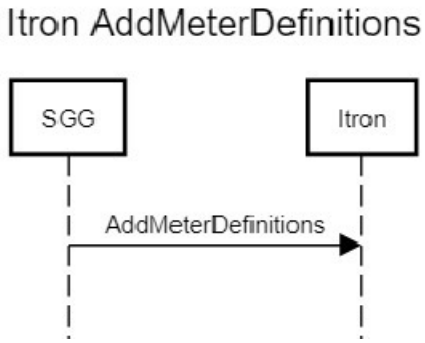
Synchronous commands, also known as “request-reply” commands, are single HTTP transactions with an initial request followed by a response. The request will contain whatever information is necessary to complete the command, such as meter identifiers or dates. The reply will contain any results generated by the command. It could be a simple acknowledgement or a complex data structure. In general, it will return quickly as HTTP transactions will fail if the response does not arrive quickly enough.

Example: Itron Commissioning Process

This section provides an example of a synchronous command, the commission device command supported by the Itron OpenWay adapter.

The Itron OpenWay 3.9 command for registering a meter (commissioning a device) in the system is AddMeterDefinitions in the `www.itron.com.ami.2008.10.provisioning.wsdl` definition. The Smart Grid Gateway Adapter for Itron OpenWay typically sends a meter serial number and some other configuration data. The response indicates whether or not an error was encountered.

The image below illustrates the process flow used by this command.



Below is a summary of the important features of the Smart Grid Gateway implementation of this command. Many of the concepts can be applied when creating a custom synchronous command.

Command Activity — Device Commissioning

Business Object: D1-DeviceCommissioning

Lifecycle and Algorithms

The table below outlines some of the important algorithms used by the Device Commissioning command activity.

Lifecycle State	Algorithms
Validate	The important features here are not the specific algorithms, but rather the ability to perform validation logic at a process level.

Lifecycle State	Algorithms
	• D1-VALDVCNAC – Validate Device Not Already Commissioned: This algorithm checks whether the device is already commissioned • D1-VHCPCOMMS – Validate Head-End's Capability to Commission Device: This algorithm checks to ensure that the head-end system has the capability to support the command requested. Some commands may not be supported by every head end system. • D1-CACTCOMM – Check for Existing Active Commissioning Command Request: This algorithm prevents multiple, simultaneous commands from being issued by SGG. • D1-CHKFDCOMM – Check for Concurrent Decommissioning Command Request: This algorithm prevents a commissioning request from executing when there is a current decommissioning request.
Commission Ready	Several algorithms in this state are designed to create outbound communications to the head end system based on various conditions. • D1-AMIOBCOMM – AMI Device Identifier Outbound Communication Creation: This algorithm creates an outbound message, a communication to the head end system.
Communication In Progress	• D1-CHKCHILD – Check Children Communications: This algorithm checks all of the outbound communications related to the activity and, if complete, will transition the activity.
Execute Completion Events	• D1-EXCMPEVTS – Execute Completion Events: This algorithm provides a mechanism for running completion events upon successful completion of the commissioning operation.

Outbound Communication — Itron - Add Meter Definition (Commission)

Business Object: D8-AddMeterDefinitionsDR

Schema

The portions of the schema that contain the payload of the web service request and response communications are the `sendDetail` element (constructed using the D8-AddMeterDefinitionsDRDA data area) and the `responseDetail` element (constructed using the D8-AddMeterDefRespDRDA data area). The XSLs used to create or decipher XML conforming to the Itron structure use these elements as a basis.

Lifecycle and Algorithms

The table below outlines some of the important algorithms used by the Add Meter Definition outbound communication.

Lifecycle State	Algorithms
Validate	• D1-VALCOMTP – Validate Communication Type: This algorithm ensures the communication type is populated on the communication.
Awaiting Response	• D8-PAMDSDDR – Itron - Populate Add Meter Definitions Send Detail - Direct Route: This algorithm constructs the `sendDetails` element used in the outbound request.

Lifecycle State	Algorithms
	• D8-COUTMCMDR – Create Outbound Message for Commissioning / Decommissioning Direct Route: This algorithm determines the correct outbound message to create based upon the input processing role. The reply from the head end system will be stored in the `responseDetails` element and log entries will be created for the creation of the outbound message and the result (whether it is a success or a failure).
Response Error	The business object will transition to this state if the reply indicates a failure. • D8-RAMBOE – Itron - Retry Add Meter BO in Error: This algorithm provides a means to retry the communication in the event of an error. • D8-CTDEBOEO – Itron - Create To Do Entry for BO in Error for Outbound: This algorithm creates a To Do entry in order to inform a user to take corrective action. • D1-ALLOWTPA – Set Allow Transition Parent Activity Flag To True For Synchronous Outbound Communication: This algorithm conditionally allows the parent activity to transition to complete.
Create Completion Event	• D8-CRCDCOMPE – Itron - Create Commission Device Completion Event: This algorithm creates a completion event to show the device as commissioned.
Completed	• D8-UCCFTPA – Itron - Update Commission Completion Flag: This algorithm notifies the parent command activity that the communication has completed.

Message Sender XSLs

Outbound messages created by Add Meter Definition outbound communication use the following XSL files:

- **Request XSL**: D8-CommissionRequest.xsl
- **Response XSL**: D8-CommissionResponse.xsl

Refer to Creating Message Senders for more information about creating message senders.

Itron Commissioning XSL Transformations

The Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay supports multiple versions of the Itron OpenWay API. To achieve this support, several of the request XSLs check a version number embedded in the source schema. Similarly, the responses XSLs are designed to handle different source XML and map the values to the common SGG format.

The Itron commissioning process uses the following XSL transformations.

D8-CommissionRequest.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:serArr="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
        xmlns:prov390="http://www.itron.com/ami/2009/08/provisioning"
        xmlns:prov370="http://www.itron.com/ami/2008/10/provisioning">

    <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes" indent="yes"/>
```

```
    <xsl:template match="/*">
        <xsl:apply-templates select="./requestEnvelope/AddMeterDefinitions"/>
    </xsl:template>

<xsl:template match="AddMeterDefinitions[../itronVersion='3.70']">
        <xsl:apply-templates select="." mode="Itr370"/>
    </xsl:template>

    <xsl:template match="AddMeterDefinitions[../itronVersion='3.90']">
        <xsl:apply-templates select="." mode="Itr390"/>
    </xsl:template>

<xsl:template match="*[local-name() != 'string']" mode="Itr370">
        <xsl:element name="{local-name()}" namespace="http://www.itron.com/ami/2008/10/
provisioning">
            <xsl:apply-templates select="*|text()" mode="Itr370"/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="*[local-name() != 'string']" mode="Itr390">
        <xsl:element name="{local-name()}" namespace="http://www.itron.com/ami/2009/08/
provisioning">
            <xsl:apply-templates select="*|text()" mode="Itr390"/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="string" mode="Itr390">
        <xsl:element name="string" namespace="http://schemas.microsoft.com/2003/10/Serialization/
Arrays">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>

<xsl:template match="string" mode="Itr370">
        <xsl:element name="string" namespace="http://schemas.microsoft.com/2003/10/Serialization/
Arrays">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>

</xsl:stylesheet>
```

D8-CommissionResponse.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:serArr="http://schemas.microsoft.com/2003/10/Serialization/Arrays"
        xmlns:prov390="http://www.itron.com/ami/2009/08/provisioning"
        xmlns:prov370="http://www.itron.com/ami/2008/10/provisioning">

    <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes" indent="yes"/>

    <xsl:template match="/*">
        <xsl:apply-templates select="./requestEnvelope/AddMeterDefinitions"/>
    </xsl:template>

<xsl:template match="AddMeterDefinitions[../itronVersion='3.70']">
        <xsl:apply-templates select="." mode="Itr370"/>
    </xsl:template>

    <xsl:template match="AddMeterDefinitions[../itronVersion='3.90']">
        <xsl:apply-templates select="." mode="Itr390"/>
    </xsl:template>

<xsl:template match="*[local-name() != 'string']" mode="Itr370">
        <xsl:element name="{local-name()}" namespace="http://www.itron.com/ami/2008/10/
provisioning">
            <xsl:apply-templates select="*|text()" mode="Itr370"/>
        </xsl:element>
    </xsl:template>
```

```
    <xsl:template match="*[local-name() != 'string']" mode="Itr390">
        <xsl:element name="{local-name()}" namespace="http://www.itron.com/ami/2009/08/
provisioning">
            <xsl:apply-templates select="*|text()" mode="Itr390"/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="string" mode="Itr390">
        <xsl:element name="string" namespace="http://schemas.microsoft.com/2003/10/Serialization/
Arrays">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>

<xsl:template match="string" mode="Itr370">
        <xsl:element name="string" namespace="http://schemas.microsoft.com/2003/10/Serialization/
Arrays">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>

</xsl:stylesheet>
```

Asynchronous Commands

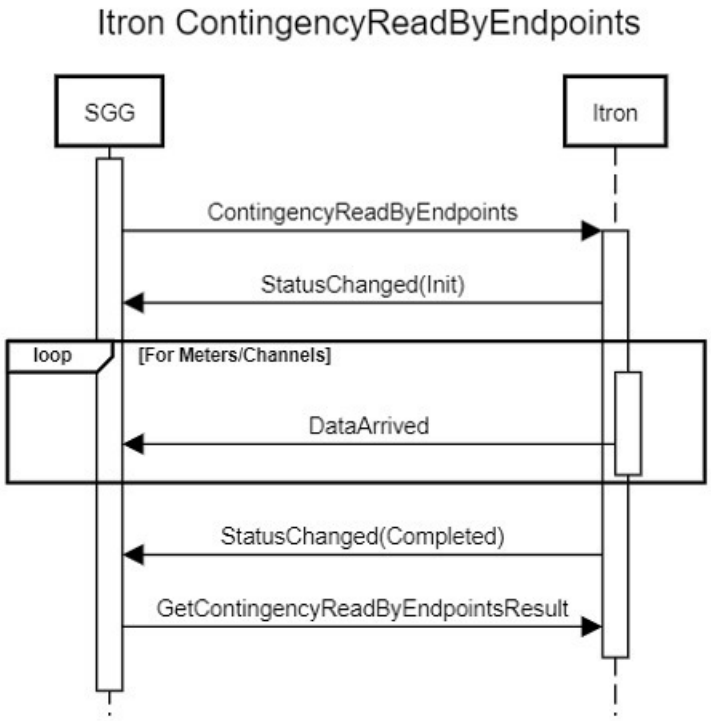
An asynchronous command is actually a collection of two or more synchronous commands. This type of command is useful when more than a few seconds will elapse before results can be collected and returned. A typical use case will begin with a Smart Grid Gateway request to start a job on the head end system. The initial request will contain enough data to initiate the process, such as a meter identifier or a date range. The response usually contains an identifier for the process started on the head end system. At some later time, a request will be initiated by the head end system. The request will contain the same process identifier so it can be linked to the original request. Some head end systems use this secondary request to indicate the status of the process. Other head end systems send the results of the process in the secondary request. Typically, the response message for either type of secondary request is a functional acknowledgement.

Example: Itron On-Demand Read Process

This section provides an example of an asynchronous command, the On-Demand Read command supported by the Itron OpenWay adapter.

The Itron `www.itron.com.ami.2008.10.data.wsdl` definition contains a series of commands for an On-Demand Read process. The Smart Grid Gateway adapter for Itron OpenWay initiates the process by invoking the ContingencyReadByEndpoints method. The reply contains a “request token ID” that identifies the entire flow and will be shared by subsequent communications. On the Itron side, a job for collecting usage and events is configured. To inform the caller (SGG) that the job is starting, Itron sends a StatusChanged request (defined in `www.itron.com.ami.2008.10.common.wsdl`) containing “State = Initialized”. Itron sends another StatusChanged request to notify SGG that the process of gathering the usage is underway. As the usage is collected, Itron submits one or more DataArrived requests to SGG. When Itron has finished making internal requests for usage, it sends a final StatusChanged request to SGG. That completion notification informs SGG that it should submit GetContingencyReadByEndpointsResult, the reply to which contains information about errors encountered during the process.

The image below illustrates the process flow used by this command.



Below is a summary of the important features of the Smart Grid Gateway implementation of this command. Many of the concepts can be applied when creating a custom synchronous command.

Command Activity — On-Demand Read (Interval)

Business Object: D1-OnDemandReadInterval

Lifecycle and Algorithms

The table below outlines some of the important algorithms used by the On-Demand Read (Interval) command activity.

Lifecycle State	Algorithms
Validate	The important features here are not the specific algorithms, but rather the ability to perform validation logic at a process level. • D1-VALDEVMC – Validate that Device has appropriate Measuring Component: This algorithm ensures the device has at least one measuring component whose Measuring Component Type is registered as interval type • D1-VALIDCMD – Validate Headend's capability to perform On Demand Read (Interval): This algorithm checks whether the head end system has the capability to support an on-demand read request. If the command is supported, an outbound communication business object is returned. • D1-CHKMST – Check for existing Measurements: This algorithm checks whether complete measurements for the specified time period exist in the database already.
Connection Ready	• D1-CODRIOB – Create Outbound communication for OnDemandRead Interval: This algorithm creates an outbound communication business object based on the Processing Role

Lifecycle State	Algorithms
	and algorithm parameters. The algorithm will look up the outbound communication business object defined for the Processing Role parameter and create an instance of that business object. In the case of this business object, the parameter is D1IN and it should create an instance of the D8-ReadIntervalDR business object. Once the Outbound Communication record is created, the Activity business object moves from the Connection Ready state to the Communication In Progress state. When the outbound communication is complete, it moves the Activity to next default state, which is Connection Ready (the activity moves from Communication In Progress *back* to Connection Ready). The outbound communication will also set the "isOnDemandReadComplete" flag value in the activity business object, in this case, indicating the outbound communication for a particular processing role was created. If the condition is true, the processing moves on to next algorithm. • D1-CODRSRTOB – Create On Demand Read Results Outbound based on Completion Flag: This algorithm is the second algorithm in this state responsible for sending outbound communications. If the completion flag passed as a parameter is true, it will create the second outbound communication. In this example, it should create an instance of D8-ReadOnDemandReadResultDR to send a second request to Itron for the results of the Contingency Read operation.
Waiting for Measurement	• D1-RIINTM – Retrieve Interval Initial Measurements: This algorithm retrieves the most recent Initial Measurement Data for each interval measuring component for the period of the measurement requested.

Outbound Communication — Itron - Contingency Read (Interval)

Business Object: D8-ReadIntervalDR

Schema

The portions of the schema that contain the payload of the web service request and response communications are the `sendDetail` element (constructed using the D8-ContingencyReadRequestDR data area) and the `responseDetail` element (constructed using the D8-ContingencyReadResponDtlDA data area). The XSLs used to create or decipher XML conforming to the Itron structure use these elements as a basis.

Lifecycle and Algorithms

The table below outlines some of the important algorithms used by the Contingency Read outbound communication.

Lifecycle State	Algorithms
Validate	• D1-VALCOMTP – Validate Communication Type: This algorithm ensures the communication type is populated on the communication.
Awaiting Response	• D8-PSDMRSCDR – Itron - Populate Send Detail for ContingencyReadByEndpoints Direct Route: This algorithm

Lifecycle State	Algorithms
	populates the `sendDetail` element with the needed items for the web service input. • D8-COUTMINDR – Itron - Create Outbound Message Contingency Read (Interval): This algorithm determines the correct outbound message to create based upon the input processing role. The reply from the head end system will be stored in the `responseDetails` element and log entries will be created for the creation of the outbound message and the result (whether it is a success or a failure).
Response Error	• D8-RBORINTER – Itron - Retry Read Interval BO in Error: This algorithm provides a means to retry the communication in the event of an error.
Completed	• D1-UCFLAGTPA – Update On-Demand Read Completion Flag And Transition Parent Activity: If requirements are met, this algorithm sets the completion flag on the parent activity to "true" and transitions it to the next state.

Inbound Communication — Itron - StatusChanged

Business Object: D8-StatusChanged

This business object handles the inbound StatusUpdate requests. It contains the request token ID created by the initial request. When it receives schema values of "State = Completed" and Result of either Canceled or Success, it triggers the next command in the Activity. For On-DemandRead, the next command is GetContingencyReadByEndpointsResult.

Schema

Unlike other communications containing `sendDetail` or `responseDetail` elements, this business object contains an input element with child elements that map to the those within the StatusChanged XSD definition.

Lifecycle and Algorithms

The table below outlines some of the important algorithms used by the Status Changed Read inbound communication.

Lifecycle State	Algorithms
Validate	• D8-IDNPOCAA – Find Parent Outbound Communication For Async Inbound: This algorithm finds the initial outbound communication associated with the current inbound communication using the request token id.
Evaluate Response	• D8-EVALSTCHG – Itron - Evaluate Response for StatusChanged: This algorithm performs a task based on the values of the arriving StatusChanged request. If the Status is Completed and the Result is Success or Failure, the parent outbound communication is completed. This alerts the activity to initiate the next outbound communication.

Outbound Communication — Itron - On Demand Read Result

Business Object: D8-ReadOnDemandReadResultDR

This is the second of two outbound requests within the Itron Contingency Read API. The outbound request submits the request token ID. The response contains error information collected by the Itron server during the execution of the Contingency Read process.

Schema

The portions of the schema that contain the payload of the web service request and response communications are the `sendDetail` element (constructed using the D8-ReadODRResultRequestDR data area) and the `responseDetail` element (constructed using the D8-ReadODRResultResponseDR data area). The XSLs used to create or decipher XML conforming to the Itron structure use these elements as a basis.

Lifecycle and Algorithms

The table below outlines some of the important algorithms used by the On-Demand Read Result outbound communication.

Lifecycle State	Algorithms
Awaiting Response	• D8-PSDODRRDR – Itron - Populate OnDemand Read Result Send Details: This algorithm populate the `sendDetail` element for the outbound message. Primarily, this consists of the request token ID. • D8-CODRROBDR – Create Outbound Message for On Demand Read Result: This algorithm determines the correct outbound message to create based upon the input Processing Role. The reply from the head end system will be stored in the `responseDetails` element and log entries will be created for the creation of the outbound message and the result (whether it is a success or a failure).
Evaluate Response	• D8-EVODRRSLT – Create Outbound Message for On Demand Read Result: This algorithm evaluates the response from Itron and determines whether the command was successful or was a failure. • D8-VDVCMCDR – Validate if Device has appropriate Measuring Component: This algorithm performs a check to ensure that the reading has a proper measuring component.

Service Script: Itron - DataArrived Request

This Groovy—based service script handles the incoming DataArrived requests. It saves the payload to Cloud Object Storage buckets based on configuration.

Schema

The schema maps closely to the Itron DataArrived message. It includes the D8-DataArrived data area.

Steps

Step 10 Edit Data – Invoke Main Groovy Method

This step shows how to trigger a Groovy script from other OUAF objects. It is similar to a main method in other programming languages. In this case, it is called by an Inbound Web Service method.

Step 20 – Groovy Imports

This step imports other classes needed by the script.

Step 30 – Groovy Members

1. Look up the Master Configuration object

2. Use the Master Configuration to find the File Storage details
    a. Find the File Adapter type
    b. Compute a filename using a configured generation script
    c. Compute a fill URL using the bucket, filename, and other details
3. Use the FileStorageAdapter business service to save the file to the targeted location.

Message Sender XSLs

Outbound messages created by on-demand read outbound communications use the following XSL files:

ContingencyReadByEndpoints

- **Request XSL**: D8-ContinegencyReadByEndpointsRequest.xsl
- **Response XSL**: D8-ContingencyReadByEndpointsResponse.xsl

GetContingencyReadByEndpointsResult

- **Request XSL**: D8-GetContingencyReadByEndpointsResultRequest.xsl
- **Response XSL**: D8-GetContingencyReadByEndpointsResultResponse.xsl

Refer to Creating Message Senders for more information about creating message senders.

Inbound Web Services

The Itron on-demand read process uses the following inbound web services.

D8-RequestStatusChangedService

- **Operation Name**: StatusChanged
- **Schema Name**: Itron — Status Changed BO
- **Request XSL**: D8-RequestStatusChangedServiceRequest.xsl
- **Response XSL**: D8-RequestStatusChangedServiceResponse.xsl

D8-DataSubscriberService

- **Operation Name**: DataArrived
- **Schema Name**: Itron — DataArrived Request Script
- **Request XSL**: D8-DataSubscriberServiceRequest.xsl
- **Response XSL**: D8-DataSubscriberServiceResponse.xsl

Itron On-Demand Read XSL Transformations

The Oracle Utilities Smart Grid Gateway Adapter for Itron OpenWay supports multiple versions of the Itron OpenWay API. To achieve this support, several of the request XSLs check a version number embedded in the source schema. Similarly, the responses XSLs are designed to handle different source XML and map the values to the common SGG format.

The Itron on-demand read process uses the following XSL transformations.

D8-ContRdByEndptsRequest.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xmlns:ns1="http://www.itron.com/ami/2008/10/data"
        xmlns:ns2="http://www.itron.com/ami/2008/10/common"
        xmlns:ns3="http://schemas.microsoft.com/2003/10/Serialization/Arrays">

    <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes" indent="yes"/>

    <xsl:template match="/*">
       <ContingencyReadByEndpoints xmlns="http://www.itron.com/ami/2008/10/data">
           <xsl:apply-templates select="./requestEnvelope/ContingencyReadByEndpoints/*"/>
       </ContingencyReadByEndpoints>
    </xsl:template>


    <xsl:template match="ElectronicSerialNumbers">
        <xsl:element name="ns2:ElectronicSerialNumbers">
            <xsl:for-each select="./string">
                <xsl:element name="ns3:string">
                    <xsl:value-of select="."/>
                </xsl:element>
            </xsl:for-each>
        </xsl:element>
    </xsl:template>


    <xsl:template match="StatusChangedService">
        <xsl:element name="ns2:StatusChangedService">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>


    <xsl:template match="Parameters">
        <ns1:Parameters>
            <ns1:ReadingStartTime><xsl:value-of select="./ReadingStartTime"/></
ns1:ReadingStartTime>
            <ns1:ReadingEndTime><xsl:value-of select="./ReadingEndTime"/></ns1:ReadingEndTime>
            <xsl:if test="string-length(./PerformDemandReset) > 0">
                <ns1:PerformDemandReset><xsl:value-of select="./PerformDemandReset"/></
ns1:PerformDemandReset>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveHomeNetworkData) > 0">
                <ns1:RetrieveHomeNetworkData><xsl:value-of select="./RetrieveHomeNetworkData"/></
ns1:RetrieveHomeNetworkData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveInstantaneousData) > 0">
                <ns1:RetrieveInstantaneousData><xsl:value-of select="./
RetrieveInstantaneousData"/></ns1:RetrieveInstantaneousData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveLastDemandReset) > 0">
                <ns1:RetrieveLastDemandReset><xsl:value-of select="./RetrieveLastDemandReset"/></
ns1:RetrieveLastDemandReset>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveLoadProfileData) > 0">
                <ns1:RetrieveLoadProfileData><xsl:value-of select="./RetrieveLoadProfileData"/></
ns1:RetrieveLoadProfileData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveLogEvents) > 0">
                <ns1:RetrieveLogEvents><xsl:value-of select="./RetrieveLogEvents"/></
ns1:RetrieveLogEvents>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveNetworkStatistics) > 0">
                <ns1:RetrieveNetworkStatistics><xsl:value-of select="./
RetrieveNetworkStatistics"/></ns1:RetrieveNetworkStatistics>
            </xsl:if>
            <xsl:if test="string-length(./RetrievePriorSelfRead) > 0">
                <ns1:RetrievePriorSelfRead><xsl:value-of select="./RetrievePriorSelfRead"/></
ns1:RetrievePriorSelfRead>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveRecentRegisters) > 0">
                <ns1:RetrieveRecentRegisters><xsl:value-of select="./RetrieveRecentRegisters"/></
ns1:RetrieveRecentRegisters>
            </xsl:if>
```

```
            <xsl:if test="string-length(./RetrieveVoltageMonitorData) > 0">
                <ns1:RetrieveVoltageMonitorData><xsl:value-of select="./
RetrieveVoltageMonitorData"/></ns1:RetrieveVoltageMonitorData>
            </xsl:if>
            <xsl:if test="string-length(./ReportSelfReadOption) > 0">
                <ns1:ReportSelfReadOption><xsl:value-of select="./ReportSelfReadOption"/></
ns1:ReportSelfReadOption>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveExtendedCurrentData) > 0">
                <ns1:RetrieveExtendedCurrentData><xsl:value-of select="./
RetrieveExtendedCurrentData"/></ns1:RetrieveExtendedCurrentData>
            </xsl:if>
            <xsl:if test="string-length(./RetrievePriorExtendedSelfReadData) > 0">
                <ns1:RetrievePriorExtendedSelfReadData><xsl:value-of select="./
RetrievePriorExtendedSelfReadData"/></ns1:RetrievePriorExtendedSelfReadData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveInstrumentationProfileData) > 0">
                <ns1:RetrieveInstrumentationProfileData><xsl:value-of select="./
RetrieveInstrumentationProfileData"/></ns1:RetrieveInstrumentationProfileData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveExtendedLoadProfileData) > 0">
                <ns1:RetrieveExtendedLoadProfileData><xsl:value-of select="./
RetrieveExtendedLoadProfileData"/></ns1:RetrieveExtendedLoadProfileData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveToolboxData) > 0">
                <ns1:RetrieveToolboxData><xsl:value-of select="./RetrieveToolboxData"/></
ns1:RetrieveToolboxData>
            </xsl:if>
            <xsl:if test="string-length(./RetrieveTemperatureMonitoringData) > 0">
                <ns1:RetrieveTemperatureMonitoringData><xsl:value-of select="./
RetrieveTemperatureMonitoringData"/></ns1:RetrieveTemperatureMonitoringData>
            </xsl:if>
        </ns1:Parameters>
    </xsl:template>


    <xsl:template match="*">
        <xsl:element name="{local-name()}" namespace="http://www.itron.com/ami/2008/10/data">
            <xsl:apply-templates select="*|text()"/>
        </xsl:element>
    </xsl:template>

</xsl:stylesheet>
```

D8-ContRdByEndptsResponse.xsl

```
<xsl:stylesheet
        version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ns1="http://www.itron.com/ami/2008/10/data"
        xmlns:com="http://www.itron.com/ami/2008/10/common"
        exclude-result-prefixes="env ns1 com">

<xsl:output method="xml" encoding="utf-8" indent="yes"/>

<xsl:template match="env:Envelope">
   <xsl:apply-templates select="./env:Body/*"/>
</xsl:template>


<xsl:template match="ns1:ContingencyReadByEndpointsResponse">
    <responseDetail>
       <responseEnvelope>
           <ContingencyReadByEndpointsResponse>
               <xsl:apply-templates select="./*"/>
           </ContingencyReadByEndpointsResponse>
       </responseEnvelope>
    </responseDetail>
</xsl:template>
```

```
<xsl:template match="env:Fault[./detail/*[contains('|OperationFault|NoValidTargetsFoundFault|
InvalidUriFormatFault|InvalidDateTimeRangeFault|ArgumentFault|TooManyConcurrentRequestsFault|
SecurityFault|', concat('|',local-name(),'|'))]]">
    <fault>
<xsl:apply-templates select="./detail/*"/>
</fault>
</xsl:template>


<xsl:template match="env:Fault">
<fault>
<xsl:apply-templates select="./*"/>
</fault>
</xsl:template>


<xsl:template match="detail">
    <detail><xsl:apply-templates select="./*" mode="serialize"/></detail>
</xsl:template>


<xsl:template match="*">
    <xsl:variable name="name">
       <xsl:value-of select="local-name()"/>
   </xsl:variable>
   <xsl:element name="{$name}" namespace="">
       <xsl:apply-templates select="*|text()"/>
   </xsl:element>
</xsl:template>


<xsl:template match="*" mode="serialize">
    <xsl:text><</xsl:text>
    <xsl:value-of select="local-name()"/>
    <xsl:for-each select="@*">
        <xsl:value-of select="concat(' ', local-name())"/>
        <xsl:text>="</xsl:text>
        <xsl:value-of select="."/>
        <xsl:text>"</xsl:text>
    </xsl:for-each>
    <xsl:text>></xsl:text>
    <xsl:apply-templates select="*" mode="serialize"/>
    <xsl:value-of select="."/>
    <xsl:text></</xsl:text>
    <xsl:value-of select="local-name()"/>
    <xsl:text>></xsl:text>
</xsl:template>

</xsl:stylesheet>
```

D8-GetContRdByEndptsRsReq.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ns1="http://www.itron.com/ami/2008/10/data"
xmlns:ns2="http://www.itron.com/ami/2008/10/common">

    <xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="yes" indent="yes"/>

    <xsl:template match="/*">
      <GetContingencyReadByEndpointsResult xmlns="http://www.itron.com/ami/2008/10/data">
         <xsl:apply-templates select="./requestEnvelope/GetContingencyReadByEndpointsResult/*" />
      </GetContingencyReadByEndpointsResult>
    </xsl:template>

    <xsl:template match="Id">
        <xsl:element name="ns2:Id">
            <xsl:value-of select="."/>
        </xsl:element>
    </xsl:template>
```

```
<xsl:template match="*">
        <xsl:element name="{local-name()}" namespace="http://www.itron.com/ami/2008/10/data">
            <xsl:apply-templates select="*|text()"/>
        </xsl:element>
    </xsl:template>

</xsl:stylesheet>
```

D8-GetContRdByEndptsRsResp.xsl

```
<xsl:stylesheet
        version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ns1="http://www.itron.com/ami/2008/10/data"
        xmlns:com="http://www.itron.com/ami/2008/10/common"
        exclude-result-prefixes="env ns1 com">

<xsl:output method="xml" encoding="utf-8" indent="yes"/>

<xsl:template match="env:Envelope">
<xsl:apply-templates select="./env:Body/*"/>
</xsl:template>

<xsl:template match="ns1:GetContingencyReadByEndpointsResultResponse">
<responseDetail>
       <responseEnvelope>
           <GetContingencyReadByEndpointsResultResponse>
               <xsl:apply-templates select="./*"/>
           </GetContingencyReadByEndpointsResultResponse>
       </responseEnvelope>
</responseDetail>
</xsl:template>

<xsl:template match="*">
<xsl:variable name="name">
        <xsl:value-of select="local-name()"/>
    </xsl:variable>
    <xsl:element name="{$name}" namespace="">
        <xsl:apply-templates select="*|text()"/>
    </xsl:element>
</xsl:template>

<xsl:template match="env:Fault[./detail/*[contains('|OperationFault|RequestNotFoundFault|
RequestNotFinishedFault|ArgumentFault|TooManyConcurrentRequestsFault|
SecurityFault|', concat('|',local-name(),'|'))]]">
    <fault>
<xsl:apply-templates select="./detail/*"/>
</fault>
</xsl:template>


<xsl:template match="env:Fault">
<fault>
<xsl:apply-templates select="./*"/>
</fault>
</xsl:template>


<xsl:template match="detail">
    <detail><xsl:apply-templates select="./*" mode="serialize"/></detail>
</xsl:template>

<xsl:template match="*" mode="serialize">
    <xsl:text><</xsl:text>
    <xsl:value-of select="local-name()"/>
    <xsl:for-each select="@*">
        <xsl:value-of select="concat(' ', local-name())"/>
        <xsl:text>="</xsl:text>
        <xsl:value-of select="."/>
        <xsl:text>"</xsl:text>
    </xsl:for-each>
```

```
    <xsl:text>></xsl:text>
    <xsl:apply-templates select="*" mode="serialize"/>
    <xsl:value-of select="."/>
    <xsl:text></</xsl:text>
    <xsl:value-of select="local-name()"/>
    <xsl:text>></xsl:text>
</xsl:template>

</xsl:stylesheet>
```

D8-DataSubSvcRequest.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
        xmlns:data="http://www.itron.com/ami/2008/10/data"
        xmlns:com="http://www.itron.com/ami/2008/10/common"
        xmlns:ev="http://www.itron.com/ami/2008/10/events"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        exclude-result-prefixes="data com ev soap11 soap12">

    <xsl:template match="soap11:Envelope">
        <xsl:apply-templates select="//data:DataArrived"/>
    </xsl:template>


    <xsl:template match="soap11:Body">
        <xsl:apply-templates select="//data:DataArrived"/>
    </xsl:template>


    <xsl:template match="data:DataArrived">
        <DataArrived>
            <input>
                <RequestToken>
                    <Id><xsl:value-of select="./data:input/data:RequestToken/com:Id"/></Id>
                </RequestToken>
                <ReadDataCollection>
                    <xsl:for-each select="./data:input/data:ReadDataCollection/data:ReadData">
                        <ReadData>
                            <Identifier><xsl:value-of select="./data:Identifier"/></Identifier>
                            <DeviceClass><xsl:value-of select="./data:DeviceClass"/></
DeviceClass>
                            <ScalingType><xsl:value-of select="./data:ScalingType"/></
ScalingType>
                            <xsl:call-template name="EventLog"/>
                            <xsl:call-template name="LoadProfileChannels"/>
                            <xsl:call-template name="NetworkStatistics"/>
                            <xsl:call-template name="RegisterValues"/>
                            <xsl:call-template name="BlockInfo"/>
                            <xsl:if test="./data:DeviceSerialNumber">
                                <DeviceSerialNumber><xsl:value-of select="./
data:DeviceSerialNumber"/></DeviceSerialNumber>
                            </xsl:if>
                        </ReadData>
                    </xsl:for-each>
                </ReadDataCollection>
                <xsl:if test="count(./data:input/data:RequestMetadata/*) > 0">
                    <RequestMetadata>
                        <xsl:copy-of select="./data:input/data:RequestMetadata/*"/>
                    </RequestMetadata>
                </xsl:if>
            </input>
        </DataArrived>
    </xsl:template>


    <xsl:template name="EventLog">
        <xsl:if test="count(./data:EventLog/ev:MeterEvents/*) > 0 or count(./data:EventLog/
ev:UnreportedEventCount) > 0">
            <EventLog>
```

```
                <xsl:if test="count(./data:EventLog/ev:MeterEvents/*) > 0">
                    <MeterEvents>
                        <xsl:for-each select="./data:EventLog/ev:MeterEvents/ev:MeterEvent">
                            <MeterEvent>
                                <Timestamp><xsl:value-of select="./ev:Timestamp"/></Timestamp>
                                <Category><xsl:value-of select="./ev:Category"/></Category>
                                <LogType><xsl:value-of select="./ev:LogType"/></LogType>
                                <Name><xsl:value-of select="./ev:Name"/></Name>
                                <xsl:if test="count(./ev:Arguments/*) > 0">
                                    <Arguments>
                                        <xsl:for-each select="./ev:Arguments/ev:Argument">
                                            <Argument>
                                                <Name><xsl:value-of select="./ev:Name"/></Name>
                                                <Value><xsl:value-of select="./ev:Value"/></
Value>
                                            </Argument>
                                        </xsl:for-each>
                                    </Arguments>
                                </xsl:if>
                                <xsl:if test="./ev:ID">
                                    <ID><xsl:value-of select="./ev:ID"/></ID>
                                </xsl:if>
                                <xsl:if test="./ev:Source">
                                    <Source><xsl:value-of select="./ev:Source"/></Source>
                                </xsl:if>
                                <xsl:if test="./ev:SequenceNumber">
                                    <SequenceNumber><xsl:value-of select="./ev:SequenceNumber"/
></SequenceNumber>
                                </xsl:if>
                            </MeterEvent>
                        </xsl:for-each>
                    </MeterEvents>
                </xsl:if>
                <xsl:if test="./data:EventLog/ev:UnreportedEventCount">
                    <UnreportedEventCount><xsl:value-of select="./data:EventLog/
ev:UnreportedEventCount"/></UnreportedEventCount>
                </xsl:if>
            </EventLog>
        </xsl:if>
    </xsl:template>


    <xsl:template name="LoadProfileChannels">
        <xsl:if test="count(./data:LoadProfileChannels/data:LoadProfileChannel/
data:IntervalLength) > 0">
            <LoadProfileChannels>
                <xsl:for-each select="./data:LoadProfileChannels/data:LoadProfileChannel">
                    <LoadProfileChannel>
                        <IntervalLength><xsl:value-of select="./data:IntervalLength"/></
IntervalLength>
                        <PulseMultiplier><xsl:value-of select="./data:PulseMultiplier"/></
PulseMultiplier>
                        <Quantity><xsl:value-of select="./data:Quantity"/></Quantity>
                        <TimeDataEnd><xsl:value-of select="./data:TimeDataEnd"/></TimeDataEnd>
                        <IntervalValues>
                            <xsl:for-each select="./data:IntervalValues/data:IntervalValue">
                                <IntervalValue>
                                    <ChannelValue><xsl:value-of select="./data:ChannelValue"/></
ChannelValue>
                                    <xsl:for-each select="./data:ProfileStatuses/
data:ProfileStatus">
                                        <ProfileStatuses>
                                            <ProfileStatus><xsl:value-of select="."/></
ProfileStatus>
                                        </ProfileStatuses>
                                    </xsl:for-each>
                                </IntervalValue>
                            </xsl:for-each>
                        </IntervalValues>
                    </LoadProfileChannel>
                </xsl:for-each>
            </LoadProfileChannels>
```

```
        </xsl:if>
    </xsl:template>


    <xsl:template name="NetworkStatistics">
        <xsl:if test="count(./data:NetworkStatistics/data:NetworkStatistic/data:Name) > 0">
            <NetworkStatistics>
                <xsl:for-each select="./data:NetworkStatistics/data:NetworkStatistic">
                    <NetworkStatistic>
                        <Name><xsl:value-of select="./data:Name"/></Name>
                        <NetworkKind><xsl:value-of select="./data:NetworkKind"/></NetworkKind>
                        <Value><xsl:value-of select="./data:Value"/></Value>
                    </NetworkStatistic>
                </xsl:for-each>
            </NetworkStatistics>
        </xsl:if>
    </xsl:template>


    <xsl:template name="RegisterValues">
        <xsl:if test="count(./data:RegisterValues/data:RegisterValue/data:Quantity) > 0">
            <RegisterValues>
                <xsl:for-each select="./data:RegisterValues/data:RegisterValue">
                    <RegisterValue>
                        <Quantity><xsl:value-of select="./data:Quantity"/></Quantity>
                        <Source><xsl:value-of select="./data:Source"/></Source>
                        <Timestamp><xsl:value-of select="substring(./data:Timestamp, 1, 19)"/></
Timestamp>
                        <Value><xsl:value-of select="./data:Value"/></Value>
                        <xsl:if test="count(./data:Status/data:string) > 0">
                            <Status>
                                <xsl:for-each select="./data:Status/data:string">
                                    <string><xsl:value-of select="."/></string>
                                </xsl:for-each>
                            </Status>
                        </xsl:if>
                    </RegisterValue>
                </xsl:for-each>
            </RegisterValues>
        </xsl:if>
    </xsl:template>


    <xsl:template name="BlockInfo">
        <xsl:if test="count(./data:BlockInfo/data:Block) > 0">
            <BlockInfo>
                <Block><xsl:value-of select="./data:BlockInfo/data:Block"/></Block>
                <TotalBlocks><xsl:value-of select="./data:BlockInfo/data:TotalBlocks"/></
TotalBlocks>
            </BlockInfo>
        </xsl:if>
    </xsl:template>


    <!-- When the SOAP Message has already been transformed by the SOAP Proxy -->
    <xsl:template match="DataArrived[namespace-uri(.) != 'http://www.itron.com/ami/2008/10/
data']">
        <xsl:copy-of select="."/>
    </xsl:template>

</xsl:stylesheet>
```

D8-DataSubSvcResponse.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:tns="http://www.itron.com/ami/2008/10/data">

    <xsl:template match="/">
        <soap11:Envelope>
            <soap11:Header/>
```

```
            <soap11:Body>
                <tns:DataArrivedResponse/>
            </soap11:Body>
        </soap11:Envelope>
    </xsl:template>

</xsl:stylesheet>
```

D8-ReqStsChgSvcRequest.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
        xmlns:com="http://www.itron.com/ami/2008/10/common"
        exclude-result-prefixes="soap11 soap12 com">

    <xsl:template match="soap11:Envelope">
        <xsl:apply-templates select="//com:StatusChanged"/>
    </xsl:template>


    <xsl:template match="soap11:Body">
        <xsl:apply-templates select="//com:StatusChanged"/>
    </xsl:template>


    <xsl:template match="com:StatusChanged">
        <StatusChanged>
            <bo/>
            <input>
                <RequestStatus>
                    <RequestToken>
                        <Id><xsl:value-of select="./com:input/com:RequestStatus/com:RequestToken/
com:Id"/></Id>
                    </RequestToken>
                    <State><xsl:value-of select="./com:input/com:RequestStatus/com:State"/></
State>
                    <EndpointsScheduled><xsl:value-of select="./com:input/com:RequestStatus/
com:EndpointsScheduled"/></EndpointsScheduled>
                    <Result><xsl:value-of select="./com:input/com:RequestStatus/com:Result"/></
Result>
                    <SuccessfulEndpoints><xsl:value-of select="./com:input/com:RequestStatus/
com:SuccessfulEndpoints"/></SuccessfulEndpoints>
                    <FailedEndpoints><xsl:value-of select="./com:input/com:RequestStatus/
com:FailedEndpoints"/></FailedEndpoints>
                    <xsl:if test="string-length(./com:input/com:RequestStatus/
com:CreatedWhen) > 0">
                        <CreatedWhen><xsl:value-of select="substring(./com:input/
com:RequestStatus/com:CreatedWhen, 1, 23)"/></CreatedWhen>
                    </xsl:if>
                    <xsl:if test="string-length(./com:input/com:RequestStatus/
com:FinishedWhen) > 0">
                        <FinishedWhen><xsl:value-of select="substring(./com:input/
com:RequestStatus/com:FinishedWhen, 1, 23)"/></FinishedWhen>
                    </xsl:if>
                    <xsl:if test="count(./com:input/com:RequestStatus/com:Metadata/*) > 0">
                        <Metadata>
                            <xsl:copy-of select="./com:input/com:RequestStatus/com:Metadata/*"/>
                        </Metadata>
                    </xsl:if>
                </RequestStatus>
            </input>
        </StatusChanged>
    </xsl:template>


    <!-- When the SOAP Message has already been transformed by the SOAP Proxy -->
    <xsl:template match="StatusChanged[namespace-uri(.) != 'http://www.itron.com/ami/2008/10/
common']">
        <xsl:copy-of select="."/>
    </xsl:template>
```

```
</xsl:stylesheet>
```

D8-RequestStatusChangedServiceResponse.xsl

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:tns="http://www.itron.com/ami/2008/10/common">

    <xsl:template match="/">
        <soap11:Envelope>
            <soap11:Header/>
            <soap11:Body>
                <tns:StatusChangedResponse/>
            </soap11:Body>
        </soap11:Envelope>
    </xsl:template>

</xsl:stylesheet>
```

Cloud Service Test Harness

The Smart Grid Gateway Cloud Service Test Harness is used to simulate a head end system for testing smart meter commands in a cloud implementation.

The test harness is "stateless", meaning it does not store meters or their current status. The harness is designed to return a successful result to all command requests by default. Devices must be created in a specific manner to return an error.

How the Test Harness Works

The test harness comprises several adapter services, each of which reside behind a "gateway" service. The gateway service is hosted on the 8080 port (by default), and when it receives requests, the gateway service examines the path and routes it to the correct SGG service running on another port. This allows all commands to use the same port, rather than specific ports for each adapter.

The individual adapter services receive their configuration from the "config" service. This monitors a directory on a file server and hosts a REST service that returns the asked-for configuration from the adapter-specific configuration file. When adapter services start up, they retrieve their configuration and then use it. The config service must be restarted for changes to the configuration file to take effect.

Creating Devices for use with the Test Harness

The test harness works with any sort of device capable to sending smart meter commands. By default, all commands are successful.

Devices must be created in a specific manner in order for errors to be returned from the test harness. Specifically, commands for devices whose identifiers (badge number, serial number, internal meter number, pallet number, external ID, MDM external ID, and asset ID) end in "ERR" will return an error. This means that when setting up devices for testing via the test harness, you must create separate devices for successful and unsuccessful (error) results.

The simplest approach when creating "error" devices is to use the same identifier value for all identifiers, but at a minimum, "ERR" MUST be the final part of the identifier value for the **AMI Device ID Type** and **AMI Pre-Commissioning Device ID Type** configured on the head end system.

Defining Head End Systems, Processing Methods, Outbound Message Types, and Message Senders for use with the Test Harness

This section provides guidelines for creating data and records used with smart meter commands when using the test harness.

Head End System

When configuring a head end system for use with the test harness, note the identifier types specified in the **AMI Device ID Type** and **AMI Pre-Commissioning Device ID Type** fields. When creating devices intended to return an error, the identifier values for the identifier type(s) specified for these fields MUST end in "ERR" to return an error from the test harness.

Outbound Message Types

When creating outbound message types for use with the test harness, use the appropriate outbound business object based on the specific command. See Smart Meter Command Flows for details.

Message Senders

When creating message senders for use with the test harness, use the following settings:

Main tab:

- **Invocation Type**: Real-Time
- **Message Class**: SOAPSNDR
- **Active**: <checked>
- **Msg Encoding**: UTF-8 message encoding

Context tab:

- **HTTP Header**: soapAction: <soapaction>

  where <soapaction> is the URI (Soap Action) for the command as shown under Adapter-Specific Test Harness Commands below.

  Note the format used here is:

  SoapAction: {URI}

  That's using a **colon, followed by a space** between the SoapAction and the value. You **cannot** use an equals (=) sign. For example:

  ```
  SoapAction: http://www.multispeak.org/Version_4.1_Release/MeterAddNotification
  ```
- **HTTP Login User**: N/A*
- **HTTP Login Password**: N/A*
- **HTTP Method**: POST
- **HTTP URL 1**: <endpoint_URL>

  where <endpoint_URL> is the HTTP URL or the command as shown under Adapter-Specific Test Harness Commands below, formatted as follows:

  ```
  http://<server_name>:<server_port>/<adapter_name>/ws/<webservice>
  ```

  where:

  - <server_name> is the machine name or name of the virtual machine where the test harness is installed
  - <server_port> is port used to run the gateway service (8080 by default)
  - <adapter_name> is one of sgg.harness.lg, sgg.harness.itron, sgg.harness.ssn, or sgg.harness.sensus, as appropriate
  - <webservice> is a name specific to each adapter (e.g. OD_Server, MR_CB, DeviceManager, and so on. See Adapter-Specific Test Harness Commands)

  *Basic Http Authentication is not enabled for the test harness. The **HTTP Login User** and **HTTP Login Password** fields need not be defined.

External Systems

When creating outbound message types on external systems and for use with the test harness, be sure to specify the appropriate Message XSL and Response XSL files based on the outbound message business object. See Creating an External System for lists of these files.

Processing Methods

Command processing methods for a head end system used with the test harness should reference an Outbound Message Type, either the **Default Outbound Message Type** or an **Outbound Message Type** for a specific **Device Type**.

Configuration Files

Each adapter service has its own configuration file that defines how the test harness operates.

Configuration files are YAML files (deployment.yaml). As of this release, configurations files are not accessible by users. Contact your Oracle Customer Success Manager to create a service request to update the configuration files used by the test harness.

See the **Configuration Files** sections under Adapter-Specific Test Harness Commands for specifics regarding the configuration file used by each adapter.

Adapter-Specific Test Harness Commands

This section provides specific details for invoking commands for specific adapters.

Each adapter comprises multiple web services that enable commands. The commands available for each web service for each adapter are provided below.

Itron OpenWay Commands

Control Service

The table below lists details for commands available via the Itron OpenWay Control Service web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Connect	ReconnectMeter	http://www.itron.com/ami/2008/10/control/ControlService/ReconnectMeter	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.control
Connect (Result)	GetReconnectMeterResult	http://www.itron.com/ami/2008/10/control/ControlService/GetReconnectMeterResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.control
Disconnect	DisconnectMeter	http://www.itron.com/ami/2008/10/control/ControlService/DisconnectMeter	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.control
Disconnect (Result)	GetDisconnectMeterResult	http://www.itron.com/ami/2008/10/control/ControlService/GetDisconnectMeterResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.control
Load Check	DetectLoadSideVoltageByMeter	http://www.itron.com/ami/2008/10/control/ControlService/DetectLoadSideVoltageByMeter	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.control

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Load Check (Result)	GetDetectLoadSideVoltageByMeterResult	http://www.itron.com/ami/2008/10/control/ControlService/GetDetectLoadSideVoltageByMeterResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.control

Control Diagnostic

The table below lists details for commands available via the Itron OpenWay Control Diagnostic web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Device Status Check	PingByEndpoints	http://www.itron.com/ami/2012/03/control/diagnostic/DiagnosticService/PingByEndpoints	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2012.03.control.diagnostic
Device Status Check (Result)	GetPingByEndpointsResult	http://www.itron.com/ami/2012/03/control/diagnostic/DiagnosticService/GetPingByEndpointsResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2012.03.control.diagnostic
Read Disconnect State	ReadDisconnectStateByMeters	http://www.itron.com/ami/2012/03/control/diagnostic/DiagnosticService/ReadDisconnectStateByMeters	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2012.03.control.diagnostic
Read Disconnect State (Result)	GetReadDisconnectStateByMetersResult	http://www.itron.com/ami/2012/03/control/diagnostic/DiagnosticService/GetReadDisconnectStateByMetersResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2012.03.control.diagnostic

Data Service

The table below lists details for commands available via the Itron OpenWay Data Service web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
On-Demand Read	ContingencyReadByEndpoints	http://www.itron.com/ami/2008/10/data/DataService/ContingencyReadByEndpoints	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.data
On-Demand Read (Result)	GetContingencyReadByEndpointsResult	http://www.itron.com/ami/2008/10/data/DataService/GetContingencyReadByEndpointsResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.data
Scheduled Read	InterrogateByGroup	http://www.itron.com/ami/2008/10/data/DataService/InterrogateByGroup	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.data
Scheduled Read (Result)	GetInterrogateByGroupResult	http://www.itron.com/ami/2008/10/data/DataService/GetInterrogateByGroupResult	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2008.10.data

Provisioning Service

The table below lists details for commands available via the Itron OpenWay Provisioning Service web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Commission	AddMeterDefinitions	http://www.itron.com/ami/2009/08/provisioning/ProvisioningService/AddMeterDefinitions	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2009.08.provisioning
Decommission	DeregisterMeters	http://www.itron.com/ami/2009/08/provisioning/ProvisioningService/DeregisterMeters	http://<server_Name>:8080/sgg.harness.itron/ws/www.itron.com.ami.2009.08.provisioning

Configuration File

The configuration file used with the Itron OpenWay adapter includes the following options:

- meter-groups
  - groups
    - [group name]: {meter1}, {meter2}, {meter3}, ...
    - [group nameERR]: {meter1}, {meter2}, {meter3}, ...
    - [default]: {meter1}, {meter2}, {meter3}, ...
- ws
  - client
    - User: An OUAF username, used for callback responses to a command
    - Pwd: An OUAF password for used callback to a command
    - callbackDelay: The number of milliseconds the test harness should wait before sending asynchronous callbacks
  - server
    - User: A Username required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
    - Pwd: A Password required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
    - validatePayload: Whether or not to perform validation against the XSD

Note: The meter-groups option is for use with the InterrogateByGroup/ScheduledRead command. It creates a list of meters which should produce usage for a given group name. If no names match the provided group name, a default set will be used. If the group name ends in "ERR," you'll get an exception.

Landis+Gyr Commands

MR_CB

The table below lists details for commands available via the Landis+Gyr MR_CB web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Commission	MeterAddNotification	http://www.multispeak.org/Version_3.0/MeterRemoveNotification	http://<server_Name>:8080/sgg.harness.lg/ws/MR_CB
Decommission	MeterRemoveNotification	http://www.multispeak.org/Version_3.0/MeterAddNotification	http://<server_Name>:8080/sgg.harness.lg/ws/MR_CB

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Meter Retire	MeterRetireNotification	http://www.multispeak.org/Version_3.0/MeterRetireNotification	\http://<server_Name>:8080/sgg.harness.lg/ws/MR_CB
On-Demand Read	InitiateMeterReadByMeterNumber	http://www.multispeak.org/Version_3.0/InitiateMeterReadByMeterNumber	http://<server_Name>:8080/sgg.harness.lg/ws/MR_CB
Scheduled Demand Reset	ScheduleDemandReset	http://www.multispeak.org/Version_3.0/ScheduleDemandReset	http://<server_Name>:8080/sgg.harness.lg/ws/MR_CB

CD_MR

The table below lists details for commands available via the Landis+Gyr CD_MR web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Meter Exchange	MeterExchangeNotification	http://www.multispeak.org/Version_3.0/MeterExchangeNotification	http://<server_name>:8080/sgg.harness.lg/ws/CB_MR

CD_CB

The table below lists details for commands available via the Landis+Gyr CD_CB web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Connect/Disconnect	InitiateConnectDisconnect	http://www.multispeak.org/Version_3.0/InitiateConnectDisconnect	http://<server_name>:8080/sgg.harness.lg/ws/CD_CB

Metering

The table below lists details for commands available via the Landis+Gyr Metering web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Add Meter to Inventory	AddMeterToInventory	turtletech.com/TS2/AddMeterToInventory	http://<server_name>:8080/sgg.harness.lg/ws/Metering

CIM

The table below lists details for commands available via the Landis+Gyr CIM web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
On Demand Read (Interval & Scalar) Device Status Check	AMIRequest	urn:LandisGyr.GSIS.CimService.CIMService/AMIRequest	http://<server_name>:8080/sgg.harness.lg/ws/CIMService

Configuration File

The configuration file used with the Landis+Gyr adapter includes the following options:

- multispeak
  - schedule-demand-reset
    - callbacks
    - -
      - source - [ip address used for callback]
      - target - [callback url]
    - -
      - source - [blank]
      - target - [callback url]
- ws
  - client
    - User: An OUAF username, used for callback responses to a command
    - Pwd: An OUAF password for used callback to a command
    - callbackDelay: The number of milliseconds the test harness should wait before sending asynchronous callbacks
  - server
    - User: A Username required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
    - Pwd: A Password required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
    - validatePayload: Whether or not to perform validation against the XSD

Note: The schedule-demand-reset key is unique to L+G ScheduleDemandReset. The L+G API contains no way to send the callback path with the initial request, so the test harness uses a configured callback path. The current approach uses the IP address of the original caller and tries to match it against one of the "source" keys in the sequence under multispeak:schedule-demand-reset:callbacks section of the configuration file. If a match is found, the associated target url is used for the callback. If the caller's URL is not found, a final "source" with a blank IP address is used as a default match and its associated "target" is used. Multiple IP address sources can be specified.

Sensus Commands

MR_Server

The table below lists details for commands available via the Sensus MR_Server web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Commission	MeterAddNotification	http://www.multispeak.org/Version_4.1_Release/MeterAddNotification	http://<server_name>:8080/sgg.harness.sensus/ws/MR_Server
Decommission	MeterRemoveNotification	http://www.multispeak.org/Version_4.1_Release/MeterRemoveNotification	http://<server_name>:8080/sgg.harness.sensus/ws/MR_Server

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
On-Demand Read	InitiateMeterReadingsByMeterNumber	http://www.multispeak.org/Version_4.1_Release/InitiateMeterReadingsByMeterID	http://<server_name>:8080/sgg.harness.sensus/ws/MR_Server

CD_Server

The table below lists details for commands available via the Sensus CD_Server web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Connect/Disconnect	InitiateConnectDisconnect	http://www.multispeak.org/Version_4.1_Release/InitiateConnectDisconnect	http://<server_name>:8080/sgg.harness.sensus/ws/CD_Server

OD_Server

The table below lists details for commands available via the Sensus OD_Server web service.

Command	Web Method	Soap Action (HTTP Header)	HTTP URL
Device Status Check	InitiateOutageDetectionEventRequest	http://www.multispeak.org/Version_4.1_Release/InitiateOutageDetectionEventRequest	http://<server_name>:8080/sgg.harness.sensus/ws/OD_Server

Configuration File

The configuration file used with the Sensus adapter includes the following options:

ws

- client
  - User: An OUAF username, used for callback responses to a command
  - Pwd: An OUAF password for used callback to a command
  - callbackDelay: The number of milliseconds the test harness should wait before sending asynchronous callbacks
- server
  - User: A Username required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
  - Pwd: A Password required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
  - validatePayload: Whether or not to perform validation against the XSD

Silver Spring Networks Commands

Note: The Silver Spring Networks adapter works differently from the other adapters. This adapter does not use SoapAction identifiers on the operations, so the HTTP Header can be left off of the Message Sender configuration.

DeviceManager

The table below lists details for commands available via the Silver Spring Networks DeviceManager web service.

Command	Web Method	HTTP URL
Commission	ReplaceLocation	http://<server_name>:8080/sgg.harness.ssn/ws/DeviceManager
Decommission	ReplaceDeviceAtLocation	http://<server_name>:8080/sgg.harness.ssn/ws/DeviceManager

JobManager

The table below lists details for commands available via the Silver Spring Networks JobManager web service.

Command	Web Method	HTTP URL
Connect/Disconnect	addRemoteProvisioningJob	http://<server_name>:8080/sgg.harness.ssn/ws/JobManager
On-Demand Read	addMeterReadJob	http://<server_name>:8080/sgg.harness.ssn/ws/JobManager
Device Status Check	addPingJob	http://<server_name>:8080/sgg.harness.ssn/ws/JobManager

DeviceResults

The table below lists details for commands available via the Silver Spring Networks DeviceResults web service.

Command	Web Method	HTTP URL
Connect/Disconnect	getRemoteProvisioningResultsByJobID	http://<server_name>:8080/sgg.harness.ssn/ws/DeviceResults
On-Demand Read	getMeterReadResultsByJobID	http://<server_name>:8080/sgg.harness.ssn/ws/DeviceResults
Device Status Check	getPingresultsByJobID	http://<server_name>:8080/sgg.harness.ssn/ws/DeviceResults

Configuration File

The configuration file used with the Silver Spring Networks adapter includes the following options:

ws

- server
  - User: A Username required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
  - Pwd: A Password required in MultiSpeakMsgHeader's UserID field. "*" means match anything. Leave blank to disable the check.
  - validatePayload: Whether or not to perform validation against the XSD

Master Configurations

Master configurations define configuration settings for individual adapters. The sections below provide details for the master configuration provided for each Smart Grid Gateway adapter.

Itron OpenWay Master Configuration

The “Itron – Master Config” master configuration defines details used by the Itron OpenWay inbound web services, including the following:

- **Command Status Changed URL Configuration**: Used to define URL configuration for StatusChangedService messages.
  - **Status Changed Service URL Script**: Used to define the service script used to derive the URL for StatusChangedService messages. When determining the URL, the system initially checks if a script has been configured on this field, and if so, that script is used. If this field is blank (which is the default), the Determine Application URL (D1-DetStChSU) script is used to determine the URL.
  - **Default Status Changed Service URL**: The default URL for “Itron RequestStatusChangedService” messages.
  - **Commands**: Command-specific URLs for “Itron RequestStatusChangedService” messages.
- **Arrived Data Save Locations**: Used to define how payloads and exceptions are routed and processed by the Itron DataSubscriberService and Itron ExceptionSubscriberService inbound web services.
  - **DataArrived Output Configuration**: Defines how payloads are routed and processed by the Itron DataSubscriberService inbound web service.
    - **File Storage**: The location where payloads are routed in the event of an error when processing the payload. This should be a File Location Extendable Lookup value.
    - **Error Output Location**: The specific bucket or folder within the file storage location where payloads with errors are routed.
    - **Processing Configuration Value**: A SGG Payload Processing Configuration extendable lookup used to define parameters for processing Itron OpenWay payloads.
  - **ExceptionsArrived Output Configuration**: Defines how exceptions are routed and processed by the Itron ExceptionSubscriberService inbound web service.
    - **File Storage**: The location where exceptions are routed in the event of an error when processing the exception. This should be a File Location Extendable Lookup value.
    - **Error Output Location**: The specific bucket or folder within the file storage location exceptions with errors are routed
    - **Processing Configuration Value**: A SGG Payload Processing Configuration extendable lookup used to define parameters for processing Itron OpenWay exceptions.

Silver Spring Networks Version Master Configuration

The “SSN – Version Master Config” master configuration specifies the version of the Silver Spring Networks head end system with which the system is communicating:

- **SSN Version**: Used to define the version of the Silver Spring Networks head end system. Options include:
  - SSN Version 4.4
  - SSN Version 4.7
  - SSN Version 4.10
- **SSN - Get Job Status Configuration**: Used to specify the **Retry Limit** and **Wait Duration (in Seconds)** for the Get Job Status requests sent by outbound communications.

Chapter 17

Defining Service Order Management Options

Understanding Service Order Management

At a high level, service order management handles requests for service as follows:

Receive / Create Service Order Request

A service order request is created and/or received. This can be as a result of a customer requesting a change to their service such as enabling or disabling service when moving into or out of a residence, but can also be the result of other business processes, such as a request to cut service due to non-payment.

Regardless of the origin of the request, a service order request is created in a customer information system (CIS) such as Oracle Utilities Customer Care and Billing), which in turn is sent to Service Order Management.

Create Service Order Activity

When Service Order Management receives the service order request, it creates a service order activity. This activity will manage and orchestrate any/all other activities needed to fulfill the original service request.

Evaluate Service Point and/or Meter

The service order activity then evaluates the current state of the service point, meter, or item, and determines the appropriate action(s) to take to fulfill the service request.

Create Activities as Needed / Appropriate

Based on the evaluation of the service point/meter/item, the service order activity creates one or more activities as needed.

- Service order field activities involve sending workers into the field to perform service. This can include meter installation, meter replacement, and other activities.
- Command activities are smart meter commands used to remotely change the state of the meter. This can include connect, disconnect, checking the device status (ping), or requesting an on-demand reading.

Following each of activities, the orchestration activity re-evaluates the state of the service point/ meter/item to determine the next appropriate action(s) required to fulfill the original service request.

For example, when enabling service at a service point with a smart scalar meter, a typical scenario might involve the following:

1. Service Order Field Activity - Install Meter
2. Command Activity: Commission Device
3. Command Activity: Remote Connect
4. Command Activity: On Demand Read - Scalar

Update Requesting System

When the service order activity determines that everything necessary to satisfy the service order request is ready, the service order activity will inform the requester and complete the original request.

Send Notification to Subscribing Systems

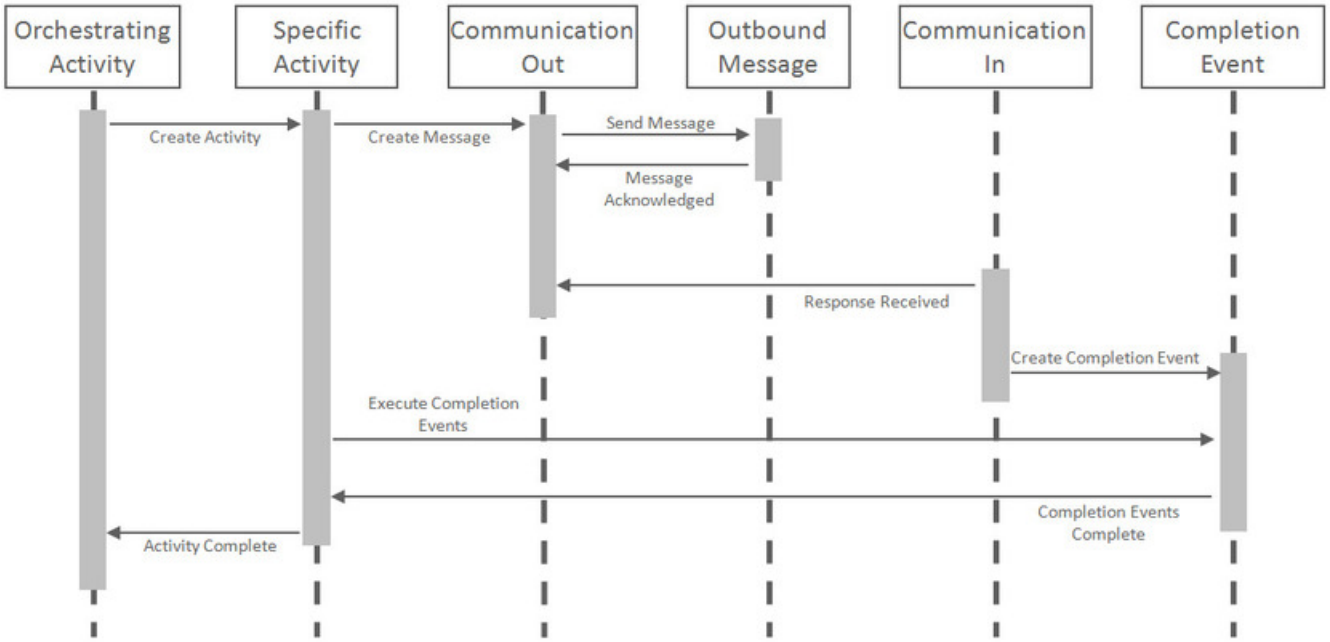
The service order activity can also be configured to send notifications to other subscribing systems regarding the status of the service point/meter/item.

Service Order Activities

Understanding Service Order Activities

This section describes service order activities and how they manage the service order process.

Service order activities coordinate a large number of child transactions that represent the communication to and from an external application. The below diagram depicts a service order activity:



Each object in the sequence diagram has a distinct set of duties within the context of the communication:

- **Orchestrating Activity**: controls the overall intent of the communication. For example it may "Enable Service" which would include initiating one-to-many specific activities to install the meter and begin the flow of the metered commodity to the service point.
- **Specific Activity**: can be initiated from an orchestrating activity or directly. These activities represent a single task to be carried out such as installing a meter or a remote disconnect smart meter command.
- **Communication Out**: orchestrates the communication to the external application and provides robust handling for any errors that might occur during that communication.
- **Outbound Message**: represents the message payload sent to the external system and the synchronous response.
- **Communication In**: orchestrates the handling of an asynchronous or unsolicited response from an external system.
- **Completion Event**: carries out the results of the communication. For example, in the case of a remote connect it would create the appropriate on off history entry for the device's installation event.

The base package provides the following types of service order activities:

- **Enable Service**: Used to enable service at a service point.
- **Disable Service**: Used to disable service at a service point.
- **Cut for Non-Payment**: Used to cut off service at a service point due to non-payment of past due amounts.
- **Reconnect Service for Payment**: Used to restore service at a service point after receipt of past due payment.
- **Exchange Meter**: Used to orchestrate the exchange of meters at service point, such as in the event that a customer upgrades their meter.
- **Back to Back Service**: Used to orchestrate a change of service when the customer at a service point changes (such as when owners/tenants change).

Service Order Activity Processing

To understand how service order activities manage the service order process, it’s important understand the lifecycle of orchestration activity business objects.

Service Order Orchestration Activity Lifecycle

All service order orchestration activity business objects share a common parent business object that defines their lifecycles. This is the Service Point Activity Orchestration business object (D1-SPActivityOrchestration). The table below outlines the lifecycle for this business object.

State	Description
Pending	The initial state for orchestration activities.
	An Enter algorithm sends an acknowledgement to the requesting system.
	The activity is transitioned to the next state via a monitor process.
Validate	Enter algorithms perform the following:
	Validate Activity Type
	Validate Service Point
	Check for a non-final duplicate service order request activity for the same service point.
Validation Error	If the business object fails any of the validations in the Pending state, it enters this state.
	Activities in this state can be corrected and retried.
Discarded	Activities discarded in other states enter this state.
	Enter algorithms perform the following:
	Validate that non-final child activities can be discarded without the need for a cancel activity

State	Description
	Cancel non-final child activities Send a failure notification to the requesting system
Waiting for Effective Date	If an orchestration activity has a future effective date, it remains in this state until the effective date is reached. A Monitor algorithm transitions the activity to the next state when the activity's effective date time is reached (process date time >= effective date time).
Are SP and Device Ready?	Each type of orchestration activity business object has a unique set of Enter algorithms that perform operations as appropriate for the type of service order request. See Service Order Activity Algorithm Types for more information about these algorithms.
Activity in Progress	Orchestration activities remain in this state while their child activities are processed. A Monitor algorithm transitions the activity to the "Are the SP and Device Ready?" state if there are no non-final child activities related to the current activity. A Monitor algorithm validates that the orchestration activity has not been in its current state for too long, based on the Expiration Days parameter on the orchestration activity's type and the Expiration Date/Time on the orchestration activity An Exit algorithm resets the Expiration Date/Time on the orchestration activity such that each time the activity exits this state its Expiration Date/Time is updated.
Activity Error	If one or more child activities enters an Error state, the orchestration activity enters this state. Activities in this state can be corrected and retried.
Retry	When an orchestration activity is retried after correction of an error condition, it enters this state. Enter algorithms perform the following: Check to determine if there are child field activities in progress that have outbound communications awaiting a response. Transition any non-final child activity to the "Reject" state (the state defined as "Reject" on the child activity business object lifecycle. This is most often the "Discarded" state).
Completed	Orchestration activities enter this state when all child activities have successfully completed. An Enter algorithm send a success notification to the requesting system.

Use the Business Object and Algorithm portals to view additional details about this business object and its lifecycle algorithms.

Cancel / Update Orchestration Lifecycle

The Cancel Orchestration (D1-CancelOrchestration) and Update Orchestration (D1-UpdateOrchestration) business objects have a similar lifecycle, with the following exceptions:

- There is no "Waiting for Effective Date" state.
- In place of the "Are SP and Device Ready?" state, they have "Cancel Specific Activity" / "Update Specific Activity" states. Enter algorithms on these states attempt to cancel or update a specific child activity.
- In the place of "Activity In Progress" and "Activity Error" states, they have "Communication in Progress" and "Communication in Error" states.

Use the Business Object and Algorithm portals to view additional details about this business object and its lifecycle algorithms.

Service Order Activity Algorithm Types

When an orchestration activity enters the "Are SP and Device Ready?" state, a set of Enter algorithms are used to evaluate the state of the service point / meter / item to determine which actions are required to complete the service request. These algorithms are based on the following algorithm types.

Customer-Device Compatibility Check (D1-DVCOMCHK)

Algorithms of this type execute the "Customer-Device Compatibility Algorithm" defined on the orchestration activity's activity type. Algorithms of this type uses the following parameters:

- **Activity BO To Create If Compatibility Detected**: Specifies the activity business object to instantiate if the algorithm detects an in compatibility between the customer/service point and device.

Note: The base package does not include algorithm types for the "Activity Type - Customer Device Compatibility" algorithm entity.

Connect Only If Previously Connected (D1-CONPRECON)

Algorithms of this type check if the "Connect New Device" flag has a value. If the flag is not populated, the algorithm sets the value of the flag based on the connection status of the device prior to the meter exchange (used only with Meter Exchange requests).

For additional information, refer to the D1-CONPRECON Algorithm Type.

Create Meter Exchange Field Activity (D1-CREMTREXC)

Algorithms of this type create a service order field activity based on details provided in the algorithm's parameters. Algorithms of this type uses the following parameters:

- **Activity and Specific Field Task to Create**: Specifies the type of activity business object and field task type to create for meter exchange service order field activities, as defined by the following mnemonics:

Mnemonic	Description
activityBOToCreate	Specifies the activity business object to create.
specificFieldTask	Specifies the Field Task Type when creating a service order field activity. This value comes from the Field Task Type extendable lookup.

For example, to create a service order field activity based on the D1-FieldActivity business object and the Exchange Meter field task type, these mnemonics would be configured as follows:

```
activityBOToCreate=D1-FieldActivity specificFieldTask=D1-ExchangeMeter
```

For additional information, refer to the D1-CREMTREXC Algorithm Type.

Decommission Removed Meter (D1-DCRMMTR)

Algorithms of this type create a decommissioning command for a removed meter (used only with Meter Exchange requests). Algorithms of this type uses the following parameters:

- **Decommission Activity BO to be created**: Specifies the type of activity business object to create when decommissioning a meter. The specific activity is created, as defined by the following mnemonics:

Mnemonic	Description
activityBOToCreate	Specifies the activity business object to create.

For example, to create a smart meter command activity based on the D1-DeviceDecommission business object, this parameter would be configured as follows:

```
activityBOToCreate=D1-DeviceDecommission
```

For additional information, refer to the D1-DCRMMTR Algorithm Type.

Create Specific Activity (D1-CRSPACT)

Algorithms of this type determine if a specific activity needs to be created based on the state of the service point. Algorithms of this type use the following parameters to specify the conditions and activity to be created:

- **Field Activity BO**: Specifies the field activity business object to instantiate if the algorithm creates a field task type (see next parameter).
- **SP State and Activity BO to Create:** Specifies the type of activity business object to create based on the state of the service point. This parameter can be repeated up to 20 times. Instances of the parameter are evaluated one at a time and the first condition matching the state of the service point is used. Parameters should be ordered from the most restrictive condition to the least restrictive. This parameters uses the following mnemonics to indicate the state (any combination of the following) of the service point:

Mnemonic	Description
servicePointConnected	Specifies if the service point is currently connected Valid values are “true" and "false".
disconnectLocation	Specifies the “Disconnect Location” for the service point. Valid values are “D1SR" (source) and "D1DV" (device).
deviceInstalledAtSP	Specifies if there is a device currently installed at the service point. Valid values are “true" and "false".
installationEventStatusOverride	Specifies the value of the "Installation Status" option type of the Install Event's Status ("Pending", "Conn-PreComm", "ManualOff", etc.)

Based on the unique combination of these mnemonics, a specific activity is created, as defined by the following mnemonics:

Mnemonic	Description
activityBOToCreate	Specifies the activity business object to create (used most often to specify a command business object)
specificFieldTask	Specifies the field task type when creating a service order field activity. This value comes from the Field Task Type extendable lookup. Note: If this mnemonic is specified, the "Field Activity BO" parameter should specify the field activity business object to create.
spTypeCategory	Specifies a service point type category. Valid values in include "D1MT" (meter), "D1IT" (item), "D1MI" (multi-item), from the SP_CATEGORY_FLG lookup. Specifying this mnemonic indicates that a service order field activity should be created only if the service point’s category match the one specified by this mnemonic.

Mnemonic	Description
executeOverrideAlgorithm	Specifies whether or not to execute the algorithm specified for the **Override Device/Task Algorithm** on the service order orchestration activity type. This allows the activity business object to create to be dynamically determined based on an algorithm instead of the "activityBOToCreate" or "specificFieldTask" mnemonics. Valid values are “true" and "false".

For example, the following parameter configuration would create a "Connect Service Point and Install Meter" service order field activity given the following conditions:

- Service Point Connected: False
- Disconnect Location: Source
- Device Installed at Service Point: False
- Service Point Category: Meter

```
servicePointConnected=false disconnectLocation=D1SR deviceInstalledAtSP=false specificFieldTask=D1-ConnSPAtSrceAndInstMtr spTypeCategory=D1MT
```

For additional information, refer to the D1-CRSPACT Algorithm Type.

Update Device (D1-UPDDVC)

Algorithms of this type determine if an activity needs to be created to update the device based on the state of the service point and device installed at the service point. Algorithms of this type use the following parameters to specify the conditions and activity to be created.

- **Error if SP Not Connected or no Device Installed (Default is Yes)**: Indicates if the algorithm should return an error if the service point is not connected or if a device is not currently installed. Valid values are "Yes" and "Con" (continue)
- **SP State and Activity BO to Create**: Specifies the type of activity business object to create based on the state of the service point. This parameter can be repeated up to 20 times. Instances of the parameter are evaluated one at a time and the first condition matching the state of the service point is used. Parameters should be ordered from the most restrictive condition to the least restrictive. This parameters uses the following mnemonic to indicate the state (any combination of the following) of the service point:

Mnemonic	Description
installationEventStatusOverride	Specifies the value of the "Installation Status" option type of the Install Event's Status ("Pending", "Conn-PreComm", "ManualOff", etc.)

Based on the value of this mnemonic, a specific activity is created, as defined by the following mnemonics:

Mnemonic	Description
activityBOToCreate	Specifies the activity business object to create (used most often to specify a command business object)
specificFieldTask	Specifies the Field Task Type when creating a service order field activity. This value comes from the Field Task Type extendable lookup. Note: If this mnemonic is specified, the "Field Activity BO" parameter should specify the field activity business object to create.

Mnemonic	Description
spTypeCategory	Specifies a service point type category. Valid values in include "D1MT" (meter), "D1IT" (item), "D1MI" (multi-item), from the SP_CATEGORY_FLG lookup. Specifying this mnemonic indicates that a service order field activity should be created only if the service point's category match the one specified by this mnemonic.
alternativeFieldTask	Specifies an alternative Field Task Type to use when creating a service order field activity in the event that the device does not support the command indicated by the "activityBOToCreate" mnemonic. Note: If this mnemonic is specified, the "Field Activity BO" parameter should specify the field activity business object to create. A value of 'skip' will continue the evaluation of the algorithm's next parameter

For example, the following parameter configuration would create a "Turn On Meter" service order field activity given the following conditions:

- Installation Status: Manual Off
- Service Point Category: Meter

```
installEventStatusOverride=ManualOff specificFieldTask=D1-TurnOnMeter spTypeCategory=D1MT
```

Other parameters used by algorithms of this type include:

- **Field Activity BO**: Specifies the field activity business object to instantiate if the algorithm creates a field task type (see above parameter).
- **XPath of Activity Element controlling Activity creation**: Defines an element within the activity business object schema that can be used to control whether or not this algorithm should create an activity. For example, to specify that the value of the "Connect New Device" flag be used to determine whether or not the algorithm should create an activity, this parameter could be set to "connectNewDevice".
- **Element value indication that Activity creation should not proceed**: Specifies a value for the element defined in the "XPath of Activity Element controlling Activity creation" parameter that would indicate that the algorithm should not create an activity. Valid values are based on the element defined for the "XPath of Activity Element controlling Activity creation" parameter. For example, to specify that an activity should not be created if the "Connect New Device" flag is set to "Do Not Connect / Turn On", this parameter should be set to "D1NC" (from the D1_CONNECT_NEW_DEVICE_FLG lookup).

For additional information, refer to the D1-UPDDVC Algorithm Type.

Remote Turn Off Turn On (D1-REMONOFF)

Algorithms of this type remotely turn a device off and on for a Back to Back service request. Algorithms of this type use the following parameters:

- **Device Incompatibility Detected Activity BO**: Specifies the activity business object the algorithm will look for. If the algorithm finds an activity of this business object, the algorithm terminates.
- **Remote Connect BO**: Specifies the activity business object to instantiate when creating a remote connect command.
- **Remote Disconnect BO**: Specifies the activity business object to instantiate when creating a remote disconnect command.

- **Installation Event Status Override for Connect**: The override status to which the Installation Event Status is set after performing a remote connect command.
- **Installation Event Status Override for Disconnect**: The override status to which the Installation Event Status is set after performing a remote disconnect command.

For additional information, refer to the D1-REMONOFF Algorithm Type.

Check for Measurement (D1-CHKMSMT)

Algorithms of this type determine if measurements exist on activity's service point as of the service date/time. If no measurement is found, algorithms of this type create an activity to either obtain or wait for a measurement. The specific type of activity is based on the type and configuration of the device and service point. Algorithms of this type use the following parameters:

- **Activity BO To Wait For Measurement:** Specifies the activity business object to instantiate when the algorithm logic indicates it should wait for a measurement for the service point.
- **Activity BO For Field Read**: Specifies the field activity business object to instantiate when the algorithm logic indicates it should request a meter reading from the field.
- **Specific Field Task**: Specifies the field task type when creating a service order field activity for a meter reading from the field.
- **Activity BO To Wait For Scheduled Read**: Specifies the activity business object to instantiate when the algorithm logic indicates it should wait for a scheduled read for the service point.
- **Activity BO For On Demand Read - Scalar**: Specifies the activity business object to instantiate when the algorithm logic indicates it should issue an on-demand read (scalar) smart meter command.
- **Start Range for Normal Measurement Condition**: The start of the range of conditions that indicate "normal" measurements when the algorithm is searching for measurements for the service point.
- **End Range for Normal Measurement Condition**: The end of the range of conditions that indicate "normal" measurements when the algorithm is searching for measurements for the service point.
- **Minimum Range for bottom Measurement condition**: The minimum measurement condition used when searching for measurements for the service point. Used only when no measurements are found in the "normal" range defined by the "Start/End Range Normal Measurement Condition" parameters.

The following parameters on the orchestration activity type are also used by algorithms of this type when searching for measurements for the service point:

- **Look for Measurement within the Day**: Limits the search to the reference date (the service date).
- **Minimum and Maximum Offset Number of Days**: Numbers of days added to /subtracted from the reference date to expand the search period.

For additional information, refer to the D1-CHKMSMT Algorithm Type.

Algorithm Types and Orchestration Activity Business Objects

Each of the orchestration activity business objects uses a different set of these algorithm types. The table below lists which of these algorithm types are defined for each of the service order orchestration activity business objects.

	Enable Service	Disable Service	Cut for Non-Payment	Reconnect for Payment	Meter Exchange	Back-to-Back
Customer-Device Compatibility Check	X					X
Connect Only If Previously Connected					X	
Create Specific Activity	X					

	Enable Service	Disable Service	Cut for Non-Payment	Reconnect for Payment	Meter Exchange	Back-to-Back
Create Meter Exchange Field Activity					X	
Update Device	X	X	X	X	X	X
Remote Turn Off Turn On						X
Decommission Removed Meter					X	
Check For Measurement:	X	X	X			X

Cancel / Update Orchestration - Algorithm Types

Enter algorithms on the "Cancel Specific Activity" and "Update Specific Activity" states attempt to cancel or update a specific child activity. These algorithms are based on the following algorithm types.

- **Cancel Specific Activity**: Algorithms of this type cancel the specific activity (either a service order field activity or a smart meter command) that is associated to the Cancel or Update orchestration activity, based on the current status of the specific activity.
- **Update Specific Activity**: Algorithms of this type update the specific activity (either a service order field activity or a smart meter command) that is associated to the Cancel or Update orchestration activity, based on the current status of the specific activity.

Algorithm Type	Algorithm(s)
Cancel Specific Activity (D1-CANSPACT)	Cancel Specific Activity (D1-CANSPACT)
Update Specific Activity (D1-UPDSPAC)	Update Specific Activity (D1-UPDSPAC)

Use the Algorithm Type and Algorithm portals to view additional details about these algorithms.

Understanding Service Order Activity Types

Service order activity types must be configured for each type of service order activity.

Service order activity types are assigned to the following Activity Type Categories:

Service Order Activity	Activity Type Category
Back-to-Back Service	Request Orchestration
Cancel Orchestration	Orchestration Maintenance
Cut Service for Non-Payment	Request Orchestration
Disable Service	Request Orchestration
Enable Service	Request Orchestration
Exchange Meter	Request Orchestration
Reconnect Service for Payment	Request Orchestration
Update Orchestration	Orchestration Maintenance

Refer to Understanding Activity Types for more information about activity types.

Configuring Service Order Activity Types

The Activity Type portal is used to display and maintain service order activity types.

Refer to Understanding Activity Types for more information.

You can access the portal from the **Admin > Communication > Activity Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Activity Type List:** This zone works differently than the typical zone that list types in that it displays both those activity types that have been configured as well as those activity types that have yet to be configured. Broadcast a record to display the details of the selected record.
- **Activity Type:** This zone provides information about the selected Activity Type

Service Order Management Configuration

The following table outlines the activity types that must be configured for the service order activity types supported by Service Order Management:

Service Order Activity	Activity Type (Business Object)
Enable Service	Enable Service (D1–EnableServiceType)
Disable Service	Disable Service (D1–DisableServiceType)
Cut Service for Non-Payment	Cut Service for Non-Payment (D1-CutServiceForNonPaymentType)
Reconnect Service for Payment	Reconnect for Payment (D1–ReconnectForPaymentType)
Meter Exchange	Exchange Meter (D1–ExchangeMeterType)
Back-to-Back Service	Back-to-Back Service (D1–BackToBackServiceType)
Cancel Orchestration	Cancel Orchestration (D1–CancelOrchestrationType)
Update Orchestration	Update Orchestration (D1–UpdateOrchestrationType)

The demonstration database contains examples of each of these service order activity types.

Service Order Field Activities

Understanding Service Order Field Activities

This section describes service order field activities and how they communicate with field work management systems.

Service order field activities are activities that involve sending workers into the field to perform service. This can include meter installation, meter replacement, and other activities.

Service order field activities send messages to a field work system, which in turn assigns them to crews to be completed in the field.

Field Activity Information

All service order field activities are based on the Field Activity (D1-FieldActivity) business object, and include the following user-accessible information:

- **Status**: The current status of the service order field activity.
- **Service Date/Time**: The date and time the service order field activity was created.
- **Service Point**: The service point associated with the service order field activity.
- **Field Task Type**: The field task type for the service order field activity. This defines the type of task and other processing details regarding how Service Order Management processes the service order field activity. See Field Task Types for more details about field task types.
- **Recipient**: The field work system service provider to which the service order field activity is sent for scheduling and assignment.
- **Device ID**: The device related to the service order field activity (if applicable).
- **Request Information**: Details of the service order request, including requester and external system information.
- **Contact Details (or Customer Information)**: Contact details for the customer associated with the service order request.
- **Address Information**: The address of the service point associated with the service order field activity.

The field activity business object also contains other information that is populated by algorithms and scripts as the service order field activity is processed by the system.

How Do Service Order Field Activities Work?

At a high level, service order field activities work as follows:

Create Field Activity

A service order orchestration activity creates a service order field activity based on the current state of the service point/meter/item.

Retrieve Required Data

The service order field activity uses a set of pre-processing algorithms to derive and populate data needed by the activity, such as the device, service point, address, effective date, and others.

Request Appointment (Optional)

If the service order field activity task type specifies that field tasks of this type require an appointment, the service order field activity checks for available appointment slots in the field work system and sends a notification to the appointment handling system.

Create Outbound Communication

The service order field activity creates an outbound communication to send the service order field activity to the field work system. The outbound communication gathers the information required by the field work system before being sent. This information is retrieved by a set of processing scripts defined on the field task type.

Receive Inbound Communication

When the service order field activity has been completed, the field work system sends an inbound communication back to Service Order Management.

Inbound communications can contain Field Activity Remarks (entered by field resources when they perform and complete their field work. If the Field Activity Remarks reference completion events, they are executed.

The inbound communications create completion events as defined on the field task type. If the service order field activity was successfully completed, it creates the "Completion Events When Successful" completion events. If the service order field activity was canceled, it creates the "Completion Events When Canceled" completion events.

Execute Completion Events

After receiving the inbound communication, a service order field activity algorithm transitions any active completion events into their executed state.

Complete Processing

The service order field activity completes its processing by doing the following:

- Updating the parent orchestration activity
- Sending a success response to the requester
- Transitioning the parent orchestration activity to the next state in its lifecycle
- Sending a service order field activity completion outbound communication to subscribing systems.

Service Order Field Activity Processing

This section outlines how service order field activities are processed.

Pre-Processing, Validation, and Post-Processing Algorithms

When service order field activities are first instantiated, a set of pre-processing algorithms populate and derive information needed for the activity, such as the activity type, service point, device, address, effective date, and other information.

Validation algorithms validate this information when first retrieved and when updated.

When service order field activities are completed, a post-processing algorithm populates the activity end date/time:

Service Order Field Activity Lifecycle

As a service order field activity moves through its lifecycle, it triggers various business processes based on the type of service order field activity. The table below outlines the lifecycle for the Field Activity (D1-FieldActivity) business object.

State	Description
Pending	The initial state for service order field activities. An Enter algorithm sends an acknowledgement to the requesting system. The activity is transitioned to the next state via a monitor process.
Validate	Enter algorithms perform the following: Validate Activity Type (and transition to error if invalid) Derive and validate service order field activity recipient Validate duplicate and conflict service order field activities Derive and validate service order field activity service point Derive and validate service order field activity device Validate address constituents Check for any existing cut service restrictions The activity is transitioned to the next state via a monitor process.
Validation Error	If the business object fails any of the validations in the Pending state, it enters this state. Enter algorithms perform the following:

State	Description
	Create a To Do based on specified To Do Type and To Do Role Set the "Allow Child to Transition Parent Activity" flag to yes. This allows the service order field activity to transition the parent orchestration activity if needed. Activities in this state can be corrected and retried.
Waiting to Request	If a service order field activity has a future effective date, it remains in this state until the effective date is reached. A Monitor algorithm transitions the activity to the next state when the activity's effective date time is reached (process date time >= effective date time). An Enter algorithm sets the "Allow Child to Transition Parent Activity Based On Effective Date" flag to yes. This allows the service order field activity to transition the parent orchestration activity if needed.
Waiting for Appointment	If the service order field activity passes its validations and the effective date has been reached, the activity enters this state. Enter algorithms perform the following: Evaluate if an appointment is required for field tasks of this type. If not, the activity transitions to the "Communication in Progress" state. Create a To Do if an appointment is necessary but the system is not able to send an appointment request Set the "Allow Child to Transition Parent Activity" flag to yes. This allows the service order field activity to transition the parent orchestration activity if needed. Send a notification to the appointment handling system Monitor algorithms perform the following: Verify if an appointment has been supplied Send a notification to the appointment handling system The activity is transitioned to the next state via a monitor process. See Waiting for Appointment for more information about this state.
Communication in Progress	Service order field activities enter this state following the "Waiting for Appointment" or "Retry" states. Enter algorithms perform the following: Create an outbound communication for the service order field activity (see Communication in Progress for more information) Set the "Allow Child to Transition Parent Activity" flag to yes. This allows the service order field activity to transition the parent orchestration activity if needed. Monitor algorithms perform the following: Check for existing child communications Check that the activity hasn't timed out
Discarded	Activities discarded in other states enter this state. Enter algorithms perform the following: Cancel outstanding outbound communications Cancel outstanding completion events

State	Description
	Populate the cancel reason Send a failure notification to the requesting system Transition the parent activity to the " Activity Error" state (see Service Order Orchestration Activity Lifecycl for more information) Check if a Cancel Orchestration activity is required
Communication Error	If an outbound or inbound communication an Error state, the service order field activity enters this state. Monitor algorithms perform the following: Check that the activity hasn't timed out Enter algorithms perform the following: Create a To Do based on specified To Do Type and To Do Role Set the "Allow Child to Transition Parent Activity" flag to yes. This allows the service order field activity to transition the parent orchestration activity if needed. Activities in this state can be corrected and retried.
Retry	When a service order field activity is retried after correction of an error condition, it enters this state. Enter algorithms perform the following: Check to determine if there are associated outbound communications in progress. Cancel any outstanding outbound communications
Execute Completion Events	After an inbound communication is received, it enters this state. Enter algorithms perform the following: Executes completion events defined on the field task type (these completion events were initially created by the inbound communication). Evaluates the "Field Activity Completed" flag on the service order field activity. If this is set to "No", the service order field activity is transitioned to the "Canceled In Field" state. The activity is transitioned to the next state via a monitor process. See Execute Completion Events for more information about this state.
Completion Event Error	If an error occurs during completion event processing, the service order field activity enters this state. Monitor algorithms perform the following: Check that the activity hasn't timed out Enter algorithms perform the following: Create a To Do based on specified To Do Type and To Do Role Set the "Allow Child to Transition Parent Activity" flag to yes. This allows the service order field activity to transition the parent orchestration activity if needed. Activities in this state can be corrected and retried.
Completed	Service order field activities enter this state when all completion events have successfully completed. Enter algorithms perform the following:

State	Description
	Update the parent orchestration activity Send a success response to the requester Transition the parent orchestration activity to the next state in its lifecycle Send a service order field activity completion outbound communication to subscribing systems.
Canceled in Field	If the "Field Activity Completed" flag on the field activity is set to "No", the service order field activity enter this state. Enter algorithms perform the following: Send a failed response to the requester Transition the parent orchestration activity to the "Activity Error" state. Create a To Do to notify users that the service order field activity has been canceled.

Waiting for Appointment

When a service order field activity enters the "Wait for Appointment" state, it first determines if an appointment is necessary for the service order field activity. If not, the activity moves on to the "Communication in Progress" state (see below).

If an appointment request cannot be sent for some reason, the service order field activity creates a To Do item to alert a user to attempt to manually request an appointment. Otherwise, the service order field activity sends an outbound message to the field work system requesting an appointment. based on the appropriate processing role defined on the."Send Notification to Appointment Handling System - Enter" algorithm.

Processing Role	Outbound Communication Business Object
Appointment Response (default) Used if: An appointment is required and needs to be scheduled Appointment has been set	Send Appointment Response Outbound Message (D1-SendApptRespOutboundMsg) Note: An outbound message must be created based on this business object.

The response from the field work system can be received by creating an Inbound Web Service that references the "Book selected appointment to Field Activity" (D1-BookAppt) service script.

While in this state, monitor algorithms verify if an appointment has been supplied and send notifications to the field work system.

Communication in Progress

Service order field activity communications are records of messages sent between Service Order Management and an external field work system. Communications can flow both outbound and inbound.

See Service Order Field Activity Communication for more information about service order field activity communication.

Execute Completion Events

After receiving the inbound communication, the service order field activity enters the "Execute Completion Events" state.

The inbound communication will have previously created completion events for the service order field activity, based on those defined on the field task type or those referenced by field activity remarks. These creation events begin in the "Pending" state.

An Enter algorithm transitions completion events associated with the service order field activity into their "Executed" state.

Service Order Field Activity Communication

This section outlines how service order field activities communicate with field work systems.

When a service order field activity enters the "Communication in Progress" state, it sends an outbound communication to the field work system, and waits for an inbound communication response.

See **Understanding the Service Order Field Activity Communication Process** below for more information about the role of communications in the service order field activity communication process.

Outbound Communications

Outbound Communications represent messages sent from Service Order Management to. an external field work system Outbound communications use the following types of objects:

Outbound Communication Business Objects

An outbound communication business object exists for each type of message to be sent to an external system. For service order field activities, the following base package outbound communication objects can be used.

Type of Outbound Communication	Outbound Communication Business Object
Initial service order field activity outbound communication	Field Activity Outbound Communication (D1-FieldActivityOBComm)
Modify outbound communication Used to send an update to a service order field activity previously sent to the field work system.	Field Activity Outbound Communication (D1-ActivityModifyOBComm)

Outbound Message Types

A outbound message type must also be created for each type of message to be sent to an external system. Again, this is based on the types of messages the system is designed to accept. For service order field activities, the following outbound message types are needed:

Type of Outbound Communication	Outbound Message Type
Initial Service Order Field Activity Message	Field Activity Outbound Message
Modify Existing Service Order Field Activity	Modify Field Activity Outbound Message

Refer to the Oracle Utilities Application Framework documentation for more information about outbound message types.

External Systems

You must also create an External System for each external system to which Service Order Management will send messages. Each external system defines a set of outbound message types that will be sent to that system. Each external system outbound message type also specifies the following:

- The processing method used to send the message (Batch, XAI, or Real-time)
- Message Sender (if Processing Method is set to Real-time or XAI)
- Batch Control (if Processing Method is set to Batch)
- Message XSL, W3C Schema, and Response XSL (as applicable)

To continue the example above, you might create the following external system:

External Application		
Outbound Message Type	**Processing Method**	**Batch Control**
Field Activity Outbound Message	Batch	Sync Request Monitor (F1-SYNRQ)
Modify Field Activity Outbound Message	Batch	Sync Request Monitor (F1-SYNRQ)

Refer to the Oracle Utilities Application Framework documentation for more information about external systems.

Inbound Communications

Inbound Communications represent messages sent from an external field work system to Service Order Management. Inbound communications are typically sent in response to a service order field activity. Inbound communications use the following types of objects:

Inbound Communication Business Objects

An inbound communication business object must be created for each type of message to be received from an external system. For service order field activities, the following base package inbound communication object can be used.

Inbound Communication Business Object
Field Activity Inbound Communication (D1-FieldActivityIBComm)

Inbound Web Service

You must also create an Inbound Web Service for each type of message to be received from an external system. Inbound web services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of inbound web services you need to create is based on the types of messages the system is designed to send. To continue the example above, you might create the following inbound web services:

Inbound Web Service	Schema (Inbound Communication Business Object)
Field Activity Inbound Communication	Field Activity Inbound Communication D1-FieldActivityIBComm

Refer to the Oracle Utilities Application Framework documentation for more information about Inbound Web Services.

Field Activity Remarks

Inbound communications can contain activity remarks, which represent notes entered by the field worker as they perform and complete their field work. These can be solely informational, or can reference completion events via the "Remark Processing" section of the Field Activity Remark Type extendable lookup. This allows information sent with the inbound communication to initiate business processing if necessary.

Completion events specified on this extendable lookup are created by the inbound communication, and then executed when the service order field activity enters the "Execute Completion Events" state.

Understanding the Service Order Field Activity Communication Process

This section provides an overview of the communication process that takes place when a service order field activity is initiated. For each step in the process, the table below provides a brief description of the processing that takes place, and lists the specific base package objects used by Service Order Management

Note that the process outlined below has been simplified for illustrative purposes, and does not reference every step performed in this process.

Step	Process	Base Package Objects
1.	An orchestration activity creates a service order field activity as part of its processing. A service order field activity business object is instantiated for the command.	Field Activity Business Object: Field Activity (D1-FieldActivity)
2.	When the service order field activity enters the Communication in Progress tate, it creates an outbound communication.	Outbound Communication Business Object: Field Activity Outbound Communication (D1-FieldActivityOBComm)
3.	A Enter algorithm on the "Awaiting Response" state of the outbound communication retrieves information needed by the outbound message to be sent to the field work system based on processing scripts specified on the field task type.	Enter Algorithm: Populate Send Detail for Field Activity (D1-POPSNDDTL)
4.	A Enter algorithm on the "Awaiting Response" state of the outbound communication creates an outbound message.	Enter Algorithm: Create Outbound Message (D1-COUTMSG) Note: An outbound message type for this message is not included in the base package.
5.	The outbound message is sent to middleware components via an External System and Batch Control. Middleware components utilize Business Process Execution Language (BPEL).	External System: MWM Batch Control: Sync Request Monitor (F1-SYNRQ)
6.	The middleware converts the outbound message from SOM format into the format used by the field work system, and sends the message to the field work system.	
7.	When the field work system sends a response, the middleware receives the response message from the field work system, and converts it from the format used by the field work system to SOM format and invokes an Inbound Web Service.	Inbound Web Service: D1-FieldActivityIBComm

Step	Process	Base Package Objects
8.	The Inbound Web Service picks up the message, and creates a corresponding inbound communication. The specific type of inbound communication business object created is determined by the Inbound Web Service.	Inbound Web Service: D1-FieldActivityIBComm Inbound Communication Business Object: Field Activity Inbound Communication (D1-FieldActivityIBComm)
9.	The inbound communication identifies the parent outbound communication.	Outbound Communication Business Object: Field Activity Outbound Communication (D1-FieldActivityOBComm)
10.	The inbound communication creates the completion events defined on the field activity field task type (Successful or Canceled, as appropriate) in the "Pending" state. If the inbound communication contains field activity remarks, it also executes any field activity remark completion events.	Inbound Communication Business Object: Field Activity Inbound Communication D1-FieldActivityIBComm
11.	The inbound communication updates the outbound communication. This update is performed by an Enter algorithm on the “Completed” Status of the inbound communication business object’s lifecycle.	Inbound Communication Business Object: Field Activity Inbound Communication D1-FieldActivityIBComm Outbound Communication Business Object: Field Activity Outbound Communication (D1-FieldActivityOBComm)
12.	The outbound communication updates the “Completion Flag” and the original service order field activity business object. This update is performed by an Enter algorithm on the “Completed” Status of the outbound communication business object’s lifecycle.	Outbound Communication BO: Initiate Connect Disconnect (D3-InitiateConnectDisconnect) Field Activity Business Object: Field Activity (D1-FieldActivity)

Unrelated Pickup Orders

When field work crews are out performing field work, it’s possible that they will encounter other work unrelated to their current task that needs to be done. This type of work can be as simple as trimming a tree whose branches are too close to power lines, or the replacement of a meter for a different customer or service point. These types of task are referred to as "unrelated pickup activities." Crews can either work the field activity or leave it to be assigned to another crew at a later date.

When the crew creates an unrelated pickup activity in the field work system, it is sent to Service Order Management, and a corresponding service order field activity is created in the system.

Unrelated pickup activities can be created via one of the following Inbound Web Services:

- Field Activity Asynchronous Req Inbound (D1-FARequestAsynchronous)
- Field Activity Synchronous Req Inbound (D1-FARequestSynchronous)

Once created, the are processed like any other service order field activity. If the pickup activity was completed in the field before being sent to Service Order Management, it will quickly move through its lifecycle (as now further action is needed) until it reaches the "Completed" state.

Retrieving Service Point Information

If the unrelated pickup activity is customer-related it will require service point information to be created. This information can be queried by the field work crew via the "Field Work Service Point Query" (D1-FieldWorkSPQuery) Inbound web service.

This service uses set of service point criteria to allow the field crew to search for a service based upon either service point or device information. The service returns a list of service points that is configurable in length. If the number of results is larger than the configured maximum length the service indicates that additional records exist and the crew can request another set of results allowing them to identify the proper service point to associate to the activity.

There are times when an unrelated pick-up activity is identified but the field crew is out-of-coverage (i.e. no network connection) and they will not be able to immediately verify service point information. In this type of situation, the crew can input the service point criteria fields and create the activity, which, when imported into Service Order Management, will attempt to identify the service point based upon the information provided. If the service point can be uniquely identified everything should operate as normal. If the service point cannot be uniquely identified then the service order field activity is set to the error state.

Understanding Service Order Field Activity Types

A single service order field activity type must be configured to support communication with external field work systems such as Oracle Utilities Module Workforce Management. Service order activity types are assigned to the “Field Activity” Activity Type Category.

Refer to Understanding Activity Types for more information about activity types.

Configuring Service Order Field Activity Types

The Activity Type portal is used to display and maintain service order field activity types.

Refer to Understanding Activity Types for more information.

You can access the portal from the **Admin** > **Communication** > **Activity Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Activity Type List:** This zone works differently than the typical zone that list types in that it displays both those activity types that have been configured as well as those activity types that have yet to be configured. Broadcast a record to display the details of the selected record.
- **Activity Type:** This zone provides information about the selected Activity Type

Service Order Management Configuration

The following table outlines the activity types that must be configured for the service order field activity type supported by Service Order Management:

Field Activity	Activity Type (Business Object)
Field Activity	Field Activity (D1–FieldActivityType)

The demonstration database contains examples of each of these service order activity types.

Field Task Types

Understanding Field Task Types

A service order field activity's field task type defines details about the type of task to be performed and how the system will process the activity.

Field Task Type Information

Field task types are values for the Field Task Type (D1-FieldTaskTypeLookup) extendable lookup. Each field task type value includes the following information:

- **Routing**: Indicates if field tasks of this type can only be performed at a service point. Valid values are "SP Required" and "Pass-Through".
- **Appointment Option**: Indicates if an appointment (via a mobile workforce application) is required or applicable to field tasks of this type. Valid values are "Not Applicable", "Optional", and "Required".
- **Completion Events When Successful**: One or more completion events that are executed upon successful completion of field tasks of this type.
- **Completion Events When Canceled**: One or more completion events that are executed upon cancellation of field tasks of this type.
- **Duplicate Task Type Information**: Defines processing rules for handling potential duplicate field tasks, including:
  - **Allow Duplicates**: Specifies whether or not duplicate field tasks are allowed
  - **Duplicate Threshold**: A number of hours used to determine if a newly instantiated field task type should be considered a duplicate.
  - **Field Task Types**: A list of one or more field task types that are considered to be duplicates of the field task type
- **Conflict Task Type Information**: Defines processing rules for handling potentially conflicting field tasks, including:
  - **Allow Conflicts**: Specifies whether or not conflicting field tasks are allowed
  - **Conflict Threshold**: A number of hours used to determine if a newly instantiated field task type should be considered a conflict.
  - **Field Task Types**: A list of one or more field task types that are considered to conflict with the field task type
- **Processing Scripts**: Defines one or more processing scripts to extract supplemental information needed by the mobile workforce application to schedule field tasks of this

Configuring Field Task Types

Field task types are configured using the extendable lookup portal.

You can access the extendable lookup portal from the **Admin > General > Extendable Lookup**.

Use the **Extendable Lookup Search** zone to search for and select the Field Task Type (D1-FieldTaskTypeLookup) extendable lookup.

The following zones may appear as part of the portal's **Main** tab page:

- **Extendable Lookup List:** This zone displays a list of values for the Field Task Type extendable lookup.
- **Extendable Lookup List:** This zone provides information about the selected value.

Service Order Management Configuration

The following table outlines the field task types that must be configured to support each of the service order activity types supported by Service Order Management:

Service Order Activity	Field Task Types
Enable Service — Meters	Connect SP at Source: D1-ConnectSPAtSource
	Connect SP at Meter: D1-ConnectSPAtMeter
	Connect SP at Source and Turn On: D1-ConnectSPAtSourceAndTurnOn
	Connect SP at Meter and Turn On: D1-ConnectSPAtMeterAndTurnOn
	Connect SP at Source and Install Meter: D1-ConnSPAtSrceAndInstMtr
	Connect SP at Meter and Install Meter: D1-ConnSPAtMtrAndInstMtr
	Install Meter: D1-InstallMeter
	Turn On Meter: D1-TurnOnMeter
Enable Service — Items	Connect SP at Source: D1-ConnectSPAtSource
	Item - Connect SP at Device: D1-ConnectSPAtDevice
	Item - Connect SP at Source and Turn On: D1-ConnSPAtSrceAndTurnOnDvc
	Item - Connect SP at Device and Turn On: D1-ConnectSPAtDvcAndTurnOn
	Connect SP at Source and Install Device: D1-ConnSPAtSrceAndInstDvc
	Connect SP at Device and Install Device: D1-ConnSPAtDvcAndInstDvc
	Item - Install Device: D1-InstallDevice
	Turn On Item: D1-TurnOnItem
Disable Service — Meters	Turn Off Meter: D1-TurnOffMeter
Disable Service — Items	Turn Off Item: D1-TurnOffItem
Cut Service for Non-Payment — Meters	Cut for Non-Payment: D1-CutForNonPayment
Cut Service for Non-Payment — Items	Item - Cut for Non-Payment: D1-CutItemForNonPayment
Reconnect Service for Payment — Meters	Reconnect for Payment: D1-ReconnectForPayment
Reconnect Service for Payment — Items	Item - Reconnect for Payment: D1-ReconnectItemForPayment
Meter Exchange	Exchange Meter: D1-ExchangeMeter
Item Exchange	Exchange Device: D1-ExchangeDevice
Back-to-Back — Meters	Read Meter: D1-ReadMeter

Service Order Management External Applications

The external systems used with Service Order Management must be defined as External Applications using the "External Application" (D1-ExternalApplication) business object. Examples of external systems can include:

- A customer information system (such as Oracle Utilities Customer Care and Billing)
- A field work system
- An asset management system (such as Oracle Utilities Operational Device Management or Oracle Utilities Work and Asset Management)

Information defined for external system service providers used by Service Order Management include:

- **Our Name/ID in Their System**: This is the value that the field work system uses to identify our system.
- **Utility Device ID Type**: This is the Device ID Type that will be used when communicating with the external application and it will be the assumed Device ID Type for any device identifiers sent from the external application.
- **Utility Service Point ID Type**: This is the Service Point ID Type that will be used when communicating with the external application and it will be the assumed Service Point ID Type for any service point identifiers sent from the external application.

Refer to Understanding External Applications and Configuring External Applications for more information about external applications.

Processing Roles

The external application's processing roles define how data relevant to the field work system is sent and/or created.

Field work service providers can use the following processing roles:

- **Activity Notification**: Used to send notifications to subscribing and/or requesting systems about the status of orchestration and/or service order field activities.
- **Appointment Request**: Used to send a request for an appointment to the field work system.
- **Cancelation Activity**: Used to send notifications to requesting systems when canceling orchestration and/or service order field activities.
- **Collection Details**: Used to retrieve details about collections processing (used with "Cut Service for Non-Payment" and "Restore Service for Payment" orchestration activities).
- **Customer Contact**: Used to send a contact to a customer regarding a service request
- **Field Activity**: Used to send a service order field activity to the field work system.
- **Field Activity Completion**: Used to send a notification regarding completion of a service order field activity.
- **Interim Status Update**: Used to send updates regarding the status of orchestration and service order field activities to requesting systems.
- **Meter Exchange Mapping**: Used to define how to define different types of meter exchanges based specific roles and device configurations. This can provide context to field crews to help ensure they install the correct type of device and device configuration when exchanging a meter.
- **Response - Appointment**: Used to send a request for an appointment to the field work system.
- **Response - Fail**: Used to send a response to an external system when Service Order Management fails to respond.
- **Response - Missed Appointment**: Used to send a response to the field work system when notification of a missed appointment is received.
- **Response - Negative Acknowledgement**: Used to send a negative acknowledgement response to an external system in the event that a request is rejected.
- **Response - Received**: Used to send a response to an external system to acknowledge receipt of a request.
- **Response - Success**: Used to send a response to an external system when Service Order Management successfully processes a request.
- **Send Field Activity Remark**: Used to send a service order field activity remark to a subscribing system
- **Update Activity**: Used to send notifications to requesting systems when updating orchestration and/or service order field activities.

Chapter 18

Defining Settlement Options

Settlement Configuration Overview

This section provides information about set up and configuration of administration data used by calculations and processes provided with Oracle Utilities Market Settlements Management.

Settlement Subscription Types

Understanding Settlement Subscription Types

Settlement subscription types are collections of properties defining a class of settlement subscriptions. Settlement subscription types also control valid values for various attributes of settlement subscriptions.

Settlement subscription types are defined by:

- **Transaction Business Object**: The business object used for settlement transactions based on settlement subscriptions of this type
- **Default Recipient**: The recipient that will be defaulted for settlement subscriptions of this type. A settlement subscription's recipient dictates the system to which results of settlement transactions are sent.
- **Transaction Retention Mode**: Defines how settlement transactions based on settlement subscriptions of this type are retained. "Keep All" indicates that all settlement transactions are retained when a replacement transaction is created (older transactions are sent to a status of "Superseded"). "Keep One Record per Period" indicates that only a single settlement transaction is retained for a time period (older transactions are deleted when the new one is completed).
- **Valid Settlement Calculation Groups**: A list of valid settlement calculation groups for settlement subscriptions of this type
- **Valid Recipients**: A list of valid recipients for settlement subscriptions of this type

- **Communication**: The outbound communication type used to create of outbound communications when sending the results of settlement transactions based on settlement subscriptions of this type

Refer to Configuring Settlement Units for more information about setting up settlement units and their respective dimension combinations, factors, and factor values.

For a deeper functional understanding, refer to the About Settlement Subscriptions or About Settlement Calculation sections of the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide*.

Configuring Settlement Subscription Types

Refer to Understanding Settlement Subscription Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Subscription Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Subscription Type List**: This zone lists all usage subscription type records. Broadcast a record to display the details of the selected record.
- **Settlement Subscription Type**: This zone displays details for the selected usage subscription type.

The settlement subscription type defines a number of lists for valid objects that can be used in conjunction with the overall usage calculation process:

- Valid Settlement Calculation Groups
- Valid Recipients

Settlement subscription types also define the outbound communication type used when sending the results of settlement transactions based on settlement subscriptions of this type.

Settlement Transaction Exception Types

Understanding Settlement Transaction Exception Types

Settlement Transaction Exception Types define the groupings of exceptions for a settlement transaction based on their functional similarity. This provides a way to define Settlement Transaction Exceptions in a distinct enough way to understand the root issue that was generated from the usage calculation rule.

For a deeper functional understanding of Settlement Calculation, refer to the About Settlement Calculations section of the *Oracle Utilities Meter Solution Business User Guide*.

Configuring Settlement Transaction Exception Types

This portal is used to display and maintain a Settlement Transaction Exception Type.

Refer to Understanding Settlement Transaction Exception Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Transaction Exception Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Transaction Exception Type List**: displays all of the Settlement Transaction Exception Types so you can choose the one you want to display in more detail

- **Settlement Transaction Exception Type**: shows the specific configuration for the selected Settlement Transaction Exception Type

Settlement Units

Understanding Settlement Units

Settlement units define the “lowest common denominator set” of aggregation dimensions. For example, if aggregations are performed based on Rate Class, Strata, Procurement Group, and Supplier, the following dimensions would form the “lowest common denominator set” of these dimensions (since these are shared by different suppliers):

- Rate Class
- Strata
- Procurement Group

Settlement units define factor values based on unique combinations of dimension values. Different factor values can be defined for each unique combination of a settlement unit’s dimensions. For example, a Rate Class, Strata, and Procurement Group settlement unit could have different factor values for each unique combination of those dimensions. These multi-variable factor values can be retrieved by value derivation algorithms when performing aggregation, estimation, and forecasting calculations based on the same dimensions as defined for the settlement unit.

For example, an aggregation measuring component that aggregates data based on a specific combination of Rate Class, Strata, and Procurement Group could retrieve a “loss” factor value from a Rate Class, Strata, and Procurement Group settlement unit based on the same combination of dimensions. See Applying Losses for an example of how settlement units can be referenced in aggregation calculations.

Aggregation processing also uses settlement units to generate counts of accounts for forecasting based on combinations of these dimensions.

Settlement units are defined by:

- **Dimensions**: The attributes by which aggregated data should be analyzed
- **Dimension Combinations**: Unique combinations of the settlement unit’s dimensions (defined in the **Settlement Unit Dimension Combination** zone)
- **Valid Factors**: Valid multi-variable factors (defined in the **Settlement Unit Valid Factors** zone)

Refer to Configuring Settlement Units for more information about setting up settlement units and their respective dimension combinations, factors, and factor values.

Configuring Settlement Units

Refer to Understanding Settlement Units for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Unit**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Unit**: This zone displays details for the settlement unit, including configured dimensions.
- **Settlement Unit Dimension Combinations**: This zone displays combinations of the settlement unit’s dimensions, based on the dimensions configured in the **Settlement Unit** zone. Click **Add Dimension Combination** in the zone’s title bar to add a new combination.

- **Settlement Unit Valid Factors**: This zone displays valid multi-variable factors for the settlement unit. Click **Add Factor** in the zone's title bar to add a new factor. Values for these factors are defined on the **Multi-Variable Factor** tab.

The following zones appear as part of the portal's **Dimension Combinations** tab page:

- **Settlement Unit Dimension Combinations**: This zone displays a list of dimension combinations for the settlement unit. Broadcast a record to display the details of the selected record. Click **Add Dimension Combination** in the zone's title bar to add a new combination.
- **Dimension Combination Values**: This zone displays factor values for each multi-variable factor defined for the selected dimension combination. You can use this zone to add and/or edit factor values.

The following zones appear as part of the portal's **Multi-Variable Factor** tab page:

- **Multi-Variable Factor List**: This zone displays a list of multi-variable factors for the settlement unit. Broadcast a record to display the details of the selected record.
- **Multi-Variable Factor Value List**: This zone displays factor values for each dimension combination for the selected multi-variable factor. You can use this zone to add and/or edit factor values.

Settlement Data Snapshot Types

Settlement data snapshots capture data as of a specific point in time for use in aggregation and settlement calculations. For instance, data snapshots can capture data such as:

- The state of a settlement account (a Settlement Item) as of the last time something about the account changed
- The state of a settlement account as of the last time the account was billed
- Measurement data used for initial settlement
- Measurement data used for final settlement

Data snapshot types define parameters for specific types of data snapshots. There are two general categories of settlement data snapshot types:

- Attribute Data Snapshot Types
- Measurement Data Snapshot Types

Attribute Groups

Understanding Attribute Groups

Attribute groups define sets of attributes used to model Settlement processing. These attributes when combined can be used to group data for aggregation, identify the appropriate estimation rules, load shapes, among other things. Attribute groups can be shared across several objects as a Settlement Item, Aggregation Measuring Component, and Settlement Unit.

Attribute groups are defined by:

- **Division**: The division to which settlement data based on attribute group belong
- **Attribute Value Business Object**: The business object used for attribute values based on the attribute group
- **Attribute Field Mapping**: Defines the fields and related extendable lookups that define the attributes defined by the attribute group. For each field configured, the related extendable lookup defines the possible values.

Refer to Configuring Attribute Groups for more information about setting up attribute groups.

Configuring Attribute Groups

Refer to Understanding Attribute Groups for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Attribute Group**.

The following zones may appear as part of the portal's **Main** tab page:

- **Attribute Group List**: This zone lists all attribute groups. Broadcast a record to display the details of the selected record.
- **Attribute Group**: This zone displays details for the selected attribute group.

Attribute Data Snapshot Types

Understanding Attribute Data Snapshot Types

Attribute data snapshots are used to capture settlement account data as of a specific point in time, such as the last time an attribute on the account changed or the last time it was billed. Attribute data snapshot types defines types of attribute data snapshots used by the system.

Attribute data snapshot types are defined by:

- **Division**: The division to which settlement account captured by attribute data snapshots of this type belong
- **Attribute Group**: The attribute group used by attribute data snapshots of this type belong
- **Attribute Data Snapshot Business Object**: The business object used for attribute data snapshots of this type
- **Processing Timetable Type**: defines the schedule for capturing attribute data snapshots of this type.
- **Attribute Field Mapping**: The specific attributes captured by attribute data snapshots of this type

Configuring Attribute Data Snapshot Types

You use the **Attribute Data Snapshot Type** portal to display and maintain attribute data snapshot types.

Refer to Understanding Attribute Data Snapshot Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Attribute Data Snapshot Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Attribute Data Snapshot Type List**: This zone lists all attribute data snapshot type records. Broadcast a record to display the details of the selected record.
- **Attribute Data Snapshot Type**: This zone displays details for the selected attribute data snapshot type.

Measurement Data Snapshot Types

Understanding Measurement Data Snapshot Types

Measurement data snapshots are used to capture measurement data as of a specific point in time, such as the when the data was used for billing. Measurement data snapshot types defines types of measurement data snapshots used by the system.

Measurement data snapshot types are defined by:

- **Division**: The division to which the measurement data captured by measurement data snapshots of this type belong.
- **Measurement Data Snapshot Business Object**: The business object used for measurement data snapshots of this type.
- **Interval Size**: The size of intervals captured by measurement data snapshots of this type (applies to interval measurement data snapshot types only).
- **Interval / Scalar**: A flag that designates the type of measurement data (interval or scalar) captured by measurement data snapshots of this type.
- **Measurement Data Source**: The source of the measurement captured by measurement data snapshots of this type.
- **Consumption Type**: The type of consumption (Billed Quantities, Metered Data, or Unmetered Data) captured by measurement data snapshots of this type (applies to scalar measurement data snapshot types only).
- **Value Identifiers**: The UOM, TOU, SQI, and short hand description for measurement data captured by measurement data snapshots of this type.

Configuring Measurement Data Snapshot Types

You use the **Measurement Data Snapshot Type** portal to display and maintain measurement data snapshot types.

Refer to Understanding Measurement Data Snapshot Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Measurement Data Snapshot Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Measurement Data Snapshot Type List**: This zone lists all measurement data snapshot type records. Broadcast a record to display the details of the selected record.
- **Measurement Data Snapshot Type**: This zone displays details for the selected measurement data snapshot type.

Settlement Calculations

Settlement Calculation Groups

Understanding Settlement Calculation Groups

Settlement calculation groups are collections of settlement calculation rules that are used to perform settlement calculations, including calculating service point quantities, totaling consumption and usage for settlement units, application of losses, and allocation of unaccounted for energy (UFE). During the settlement transaction process, the system executes the settlement calculation rules defined in the settlement calculation group referenced on the settlement subscription. The rules within a settlement calculation group are defined in a specific sequence, allowing control over the order in which the rules are executed.

Settlement calculation groups are associated with specific settlement subscriptions and settlement subscriptions types (or both). When assigned to settlement subscriptions, settlement calculation groups contain the settlement calculation rules used to perform the appropriate calculations. Settlement calculation groups associated with settlement subscription types are those groups considered valid for settlement subscriptions of that type.

Settlement calculation groups can also be referenced by the Execute Usage Calculation Group calculation rule.

Understanding Bulk Settlement Calculation Groups

Bulk settlement calculation groups are calculation groups intended for use with high volume calculations, such as UFE allocation.

Some specific differences between bulk settlement calculation groups and standard settlement calculation groups include:

- Bulk settlement calculation groups reference only a single settlement calculation rule.
- Execution of bulk settlement calculation groups is performed by batch execution only, typically within a set of calculations performed during settlement transaction processing.
- Bulk settlement calculation groups use the “Settlement Bulk Calculation Group” business object (D1–SettlementBulkCalcGroup).
- Bulk settlement calculation groups have a **Calculation Group Class** of “Settlement Bulk Calculation”.

Configuring Settlement Calculation Groups

This portal is used to display and maintain a settlement calculation group.

Refer to Understanding Settlement Calculation Groups for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Calculation Group**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Calculation Group**: This zone displays basic information about a settlement calculation group.
- **Settlement Calculation Rules List**: This zone lists the settlement calculation rules belonging to the group. Not applicable to bulk calculation groups.
- **Referencing Settlement Calculation Rules List**: This zone lists the settlement calculation rules that reference the group. Not applicable to bulk calculation groups.
- **Referencing Settlement Subscription Type List**: This zone lists the settlement subscription types that reference the group.
- **Referencing Settlement Subscription List**: This zone lists the settlement subscriptions that reference the group.

Configuring Bulk Settlement Calculation Groups

The **Settlement Calculation Group** portal is used to display and maintain bulk settlement calculation groups.

Refer to Understanding Bulk Settlement Calculation Groups for more information about bulk settlement calculation groups and Configuring Settlement Calculation Groups for more information about creating and maintaining bulk settlement calculation groups.

The following apply when creating new bulk settlement calculation groups:

- Select “Settlement Bulk Calculation Group” from the **Settlement Calculation Group Business Object** drop-down list.
- Use a naming convention of some sort to identify bulk settlement calculation groups (the **Settlement Calculation Group** portal does not provide a way to search for groups based on **Calculation Group Class**). For instance, consider including “bulk” or “blk" in the **Description** of bulk settlement calculation groups.

Settlement Calculation Rules

Understanding Settlement Calculation Rules

Settlement calculation rules are standard and custom rules that perform settlement calculations, including calculating service point quantities, totaling consumption and usage for settlement units, application of losses, and allocation of unaccounted for energy (UFE). Settlement calculation rules are created for a specific settlement calculation group. For example, if you were configuring two settlement calculation groups and both included a specific usage calculation rule, you would need to create two instances of the settlement calculation rule, one for each group.

On almost every settlement calculation rule, the failure of the rule results in a settlement transaction exception and the Usage Transaction Exception Type for the failure can be configured on the rule. These usage transaction exception types can also be set to a specific Exception Severity:

- **Information**: Used to highlight minor issues, but not sufficient to cause the settlement transaction to be put into a failure state. Exceptions of this category can be used to report on the frequency of interesting, but not fatal issues.
- **Issues**: Used to report a problem that will prevent the settlement transaction from being sent. Multiple "issue exceptions" can be created during settlement transaction processing. If at least one issue exists after all rules have been applied, the settlement transaction is transitioned to a failure state requiring review and approval.
- **Terminate**: Used to report a severe issue that will cause the settlement calculation process to stop and the settlement transaction to be transitioned immediately to a failure state requiring review and approval. Only one terminate exception can be issued (as the first one causes calculation processing to stop for a settlement transaction). This should be used for cases where manual override / approval isn't accurate.

For a deeper functional understanding of settlement calculation, refer to the About Settlement Calculation section of the *Oracle Utilities Meter Data Management / Smart Grid Gateway Business User Guide*.

Configuring Settlement Calculation Rules

This portal is used to display and maintain a settlement calculation rule.

Refer to Understanding Settlement Calculation Rules, Understanding Settlement Calculation Groups, and Understanding Usage Transaction Exception Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Calculation Rule**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Calculation Rule**: This zone displays details for the selected settlement calculation rule, including parameters used when executing the rule
- **Eligibility Criteria List**: Lists the eligibility criteria defined for the rule, if applicable.

Calculation Settlement Calculation Rules

The following is a list of the calculation settlement calculation rules provided as part of the base product. For more information on how each rule executes and can be configured, follow the link provided on the rule.

Settlement Calculation Rule Name	Applicable Data Type(s)	Purpose
Apply Math (Interval Data)	Interval	This rule is used to perform calculations on interval data and stores the results in the

		settlement transaction's service quantities. A variety of options are available on this rule that include defining the calculation type, variables to use, as well as the equation to use (math functions and expressions). This rule provides aggregated usage for all selected interval measuring components (filter by TOU, SQI & UOM) associated to a settlement subscription. This rule can also multiply total usage by a factor using a custom formula.
Array Math	Interval, arrays, and calculated quantities	This rule is used to perform settlement calculations on arrays and interval data and it stores the results in the settlement transaction's service quantities.
Market Award Allocation	Interval	This rule is used to perform settlement calculations to allocate load based on market awards, based on appropriate methods for dividing load out to wholesale contracts.
Measuring Component Set Calculation	Interval and calculated quantities	This rule is used to perform UFE calculations.
Vector and Service Quantity Math	Interval	This rule is designed to facilitate configuration of complex vector calculations. It is based on a series of underlying services with vectors configured as input to the calculations. Typical uses: Perform math using interval data. For example, take the difference between two curves, find max values, find coincident peaks, multiply a curve by a value, apply TOU maps, etc. Define complex formulas using various interval curves, profile factor values or calculated service quantities (bill determinant values). Support math functions: sin, cos, square root, etc. Store derived curves in memory that can be used in subsequent calculations Please note, this rule is not as efficient as other rules.

Decision-Making Settlement Calculation Rules

There are settlement calculation rules delivered as part of the base product that help with decision-making when executing the settlement calculation process. For more information on how each rule executes and can be configured, follow the link provided on the rule.

Settlement Calculation Rule Name	Purpose

Execute Usage Calculation Group	This rule performs a call to execute a separate settlement calculation group which includes execution of all settlement calculation rules within that group.
Exception Handler	This rule is used to terminate processing if exception count criteria specified in the rule is met.

Advanced Aside: Using Factors For Variables

A situation common in some implementations involves converting one unit of measure (UOM) to another. However, the conversion factor used in conversions of this type can differ based on many different types of criteria, such as the location of the service point or other characteristics. This type of calculation can be implemented as a settlement calculation rule that accumulates consumption for one UOM and converts the consumption to a different UOM by applying a factor to it.

Factors used for this purpose have a Factor Class of "Number," and use some unique rules:

- Number factors reference a characteristic type (with pre-defined values).
- Number factors reference an algorithm that retrieves or derives the value of the characteristic type at runtime.

Factor values for a Number factor are effective-dated pairings of a characteristic value and a corresponding value. Because these pairings are effective-dated, the value returned from the factor can change over time for each characteristic value. At run time, the rule retrieves / derives the characteristic value for the factor's characteristic type and then finds the value associated with the respective characteristic value. Factors can be related to any real or dynamic attribute, so rules of this type are very flexible. For example:

- **Real Attribute**: you could create a rule that retrieves a specific value based on the location of a service point.
- **Dynamic Attribute**: you could create a rule that retrieves a percentage value based on the amount the customer conserved as compared to the same period in the prior year, returning one value if the amount conserved is between 5% and 10%, another value if the amount conserved is between 10% and 20%, and a third value if the amount conserved is greater than 20%. The amount conserved is dynamically calculated at execution time and is compared to the characteristic values defined for the factor, and returns the appropriate value. In this example, if the amount conserved was anything less than 5%, no percentage value would be returned.

Calculation Rules

Array Math

This rule is used to perform settlement calculations on arrays and interval data and it stores the results in the settlement transaction's service quantities. Arrays can be derived from either measuring component sets or calculated quantities. Interval data vectors can be derived from measuring component sets, individual measuring components, or calculated quantities. A variety of options are available on this rule that include defining variables to use for arrays and interval data vectors, the specific calculations to perform (leveraging math functions and expressions), and how to save the results.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-ARRAYMATH Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-ArrayMath

Market Award Allocation

This rule is used to perform settlement calculations to allocate load based on market awards, based on appropriate methods for dividing load out to wholesale contracts. This rule takes in a vector from the settlement transaction and divides the load to be allocated to various market awards for a configured market product set. The allocated amounts should be tied to the contract connected to the market award and then saved to the settlement transaction as a new array.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D2-AWARDALOC Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-MarketAwardAllocation

Measuring Component Set Calculation

This rule is used to perform settlement UFE calculations.

This rule executes a batch program (D1-MCSCA, defined as a parameter on the Execute MC Set Calculation-Apply Rule algorithm), which in turn executes the Perform MC Set Calculation algorithm to perform the calculation. The rule configuration includes definition of variables and the equation used for UFE calculations. Variables can be based on specified measuring component sets, aggregation measuring components, or calculated quantities.

This rule should be used ONLY with Bulk Settlement Calculation Groups.

NOTE:
Additional detail on the logic of this rule can be found in the Detailed Description of the D1-EXEMCSCAL Algorithm Type.

For help with the meaning of specific configuration fields, refer to the embedded help on the screen when adding or editing the rule.

Business Object

D2-MCSetCalculation

Settlement Item Types

Understanding Settlement Item Types

Settlement Item Types define specific types of end-customer settlement accounts.

Settlement item types define the following attributes:

- **Settlement Item Business Object**: The business object to use create settlement items of this type.
- **Settlement Billed Usage Business Object**: The business object to use to capture billed usage for a settlement account.
- **Valid Service Point Types**: Valid service point types for the settlement item type.

NOTE: Settlement item types are similar to usage subscription types used in Meter Data Management, and are based on the Usage Subscription Type maintenance object (D1-USTYPE).

Configuring Settlement Item Types

Refer to Understanding Settlement Item Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Item Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Item Type List**: This zone lists all settlement item types. Broadcast a record to display the details of the selected record.
- **Settlement Item Type**: This zone displays details for the selected settlement item type.

Settlement Item Quantity Types

Understanding Settlement Item Quantity Types

Settlement item quantity types define types of quantities that can be stored for a settlement item. These quantity types are often infrequently calculated descriptors of a given customer and how they relate, at an aggregate level, to either the customer base as a whole or their particular rate class.

Settlement item quantity types use the following parameters:

- **Service Type**: The service type (electric, gas, water, etc.) for settlement item quantities of this type.
- **Settlement Item Quantity Business Object**: The business object used for settlement item quantities of this type.
- **Settlement Item Quantity Identifiers**: Value identifiers related to the current settlement item quantity type (used to provide shorthand descriptions of the various types of values measured by settlement item quantities of this type).

Configuring Settlement Item Quantity Types

You use the **Settlement Item Quantity Type** portal to display and maintain settlement item quantity types.

Refer to Understanding Settlement Item Quantity Types for more information.

You can access the portal by selecting **Admin**, then **Settlement**, then **Settlement Item Quantity Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Settlement Item Quantity Type List**: This zone lists all settlement item quantity type records. Broadcast a record to display the details of the selected record.
- **Settlement Item Quantity Type**: This zone displays details for the selected settlement item quantity type.

Market Contract Types

Understanding Market Contract Types

Market Contact Types define specific types of market contracts used in settlement processing.

Market contract types define the following:

- The market associated with market contracts of this type
- The pricing category for market contracts of this type
- Whether market contracts of this type have Unique Contract Buyers. When set to "Yes", market contracts of this type are validated to ensure that each buyer only has one valid contract at a point in time for the market contract type.

Configuring Market Contract Types

Refer to Understanding Market Contract Types for more information.

You can access the portal by selecting **Admin**, then **Market**, then **Market Contract Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Market Contract Type List**: This zone lists all market contract type records. Broadcast a record to display the details of the selected record.
- **Market Contract Type**: This zone displays details for the selected market contract type.

Market Products

Market Product Sets

Understanding Market Product Sets

Market Product Sets define the highest level grouping for a set of products that will be processed together. Market Product Sets are used by the Market Award Allocation settlement calculation rule as a way to gather a set of products and their related awards for division of load into an array of interval curves / vectors.

Market product sets define the following:

- The market associated with the product set
- The service type (electric, gas, etc.) of the product set

Configuring Market Product Sets

Refer to Understanding Market Product Sets for more information.

You can access the portal by selecting **Admin**, then **Market**, then **Market Product Set**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Market Product Set**: This zone displays details for the selected market product set.

Market Product Types

Understanding Market Product Types

Market Product Types define types of products used in settlement processing.

There are two primary types of market product types, each based on a specific business object:

- **Market Product Type**: Used for market product types whose award allocation is based on the configuration of the Market Award Allocation settlement calculation rule.
- **Formula-Based Market Type**: Used for market product types with more complex award allocation. This business object provides the ability to configure complex conditional calculations that take in both interval and scalar variables.

Market product types are defined by the following:

- **Division**: The division associated with the market product type.
- **Number of Decimals**: The number of decimal values used for market award quantities based on market products of this type.
- **Product Business Object**: The market product business object used for market products of this type.
- **Award Business Object**: The market award business object used for market awards based on market products of this type.
- **Calculation Details**: Applicable to Formula-Based Product Types only. Details used in performing formula-based calculations for market products of this type. Details include:
  - **Calculation Pre-Processing**: One or more algorithms to be executed prior to performing calculations for products of this type. These can include validation algorithms used to ensure the data being used by the calculations is valid and algorithms that retrieve, calculate, and store values used later in the calculation.
  - **Calculation Inputs**: Input variables (including Fixed Variables, Award Adhoc Characteristics, Factor Numbers, and Calculated Quantities) and Input Vectors (Fixed Vector Variables, Calculated Quantities, and Factor Profiles) that will be used in the calculations performed for products of this type.
  - **Calculations**: One or more calculations (based on the Calculation Inputs) to be performed for products of this type. Calculations can be based on Simple or Conditional formulas.
  - **Calculation Post-Processing**: Details of how the results of the calculations are saved and used in later processing. Options include:
    - **Save Results to Calculated Quantity**: Defines how calculation results are saved as service quantities on the Settlement Transaction that initiated the market product calculation and, if applicable, assigned to market contracts.
    - **Save Results to Characteristics**: Defines how calculation results are saved as characteristic values on the Settlement Transaction that initiated the market product calculation.

Configuring Market Product Types

Refer to Understanding Market Product Types for more information.

You can access the portal by selecting **Admin**, then **Market**, then **Market Product Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Market Product Type List**: This zone lists all market product type records. Broadcast a record to display the details of the selected record.
- **Market Product Type**: This zone displays details for the selected market product type.

Settlement Module Configuration

Oracle Utilities Market Settlements Management shares its application environment with other Oracle Utilities meter solution products, including Oracle Utilities Meter Data Management and Oracle Utilities Smart Grid Gateway. The default installation of any of these products provides access to all menu items and functionality provided with these products.

To configure the system such that only menu items and functionality used by Oracle Utilities Market Settlements Management is available, application users should be assigned to the following set of user groups:

- Meter Stack Common Services
- Meter Stack Common Batch Services*
- Settlement Services
- Settlement Batch Services*

*These user groups should be restricted to only those users who will be executing batch processing.

Refer to Defining Users in the *Oracle Utilities Application Framework Administrative User's Guide* for more information about setting up users.

Chapter 19

Additional Independent Modules

Aggregation

Standard Aggregation

Configuring an Out-of-the-box Aggregation

Aggregation calculations should be run on an as needed basis. This can include running the following batch processes:

- Scanning for new aggregation dimension (D1-ADS): This process is applicable if the system is configured to use aggregation dimension scanners to detect new aggregation dimensions (such as a service point referencing a new transformer for which an aggregator measuring component doesn't currently exist)
- Performing aggregation calculations (D2-AGG): this batch handles the summarization of measurement values in order to create aggregated measurements.

Note that aggregation calculations should precede usage transaction processing if aggregated values serves as input to the calculation of bill determinants.

Refer to the About Aggregations section of the *Oracle Utilities Meter Solution Business User Guide* for more information on this functionality. Refer to Understanding Measuring Component Types for information on Measuring Component Types provided for aggregation.

Understanding an Example Out-of-the-box Aggregation

The Oracle Utilities Meter Data Management base package includes an aggregation that aggregates measurement quantities for constituent measuring components based on postal code and service type dimensions. The table below outlines the types of objects used in this aggregation, based on the steps outlined above), and the specific objects for each type.

Dependency Order	Object Type	Base Package Example
1	Aggregator Measuring Component Business Object	D2-Aggregator (Aggregator - Postal and Service Type)
2	Aggregator Measuring Component UI Maps	Display: D2-AggMCDisp (Service Type and Postal Aggregator-Display) Maintenance: D2-AggMCMaint (Service Type and Postal Aggregator-Maintenance)
3	Aggregator Business Object Info Algorithm	D2-AMC-INFO (Service Type and Postal Aggregator - Information)
4	Find Constituent Measuring Components Algorithm	D2-DET-CMC (Find Constituent Measuring Components Based on Service Type and Postal)
5	Measuring Component Type	D2-AggregatorType (Aggregator Type)
6	Query Zone for Consumption Statistics Portal	D2-AGGMCQRY (Aggregator Search)
7	Dimension Scanner Activity Business Object	D2-ActivityAggDimScanner (Aggregator Creator - Postal / Service Type)
8	Dimension Scanner Activity UI Maps	Display: D2-AggDimScannerActDisp (Aggregator DS Activity-Display) Maintenance: D2- AggDimScannerActMaint (Aggregator DS Activity-Maintenance)
9	Dimension Scanner Activity Business Object Info Algorithm	D2-ADS-INFO (Aggregator Dimension Scanner Information)
10	Enter Algorithm for Scan State	D2-CRE-AGGMC (Aggregator MC Creation for Post Code and Service Type)
11	Activity Type	D2-ActivityTypeAggDimScanner (Aggregator Dimension Scanner Activity Type)

Refer to the About Aggregations section of the *Oracle Utilities Meter Solution Business User Guide* for more functional information.

Creating a New Custom Aggregation

This section describes the overall process for creating a new custom Aggregation.

Refer to the About Aggregations section of the *Oracle Utilities Meter Solution Business User Guide* for more functional information.

Execute the following steps:

1. Create a business object for the aggregator measuring component. This will flatten the dimensional value(s) into searchable characteristics. Whether this business object is a parent or a child of another aggregator business object depends on when periodic aggregation should occur:
   a. If you want the periodic aggregation to occur when another aggregation occurs, it can be a child business object (meaning that it inherits the lifecycle (and therefore the deferred monitor) of the parent)
   b. If you want to schedule its periodic aggregation independently from other aggregation business objects, this must NOT be a child business object as it will require its own deferred monitor (and deferred monitors can only be defined on parent business objects)
2. Create UI maps for the aggregator business object as follows:
   a. One to display the aggregator measuring component (Display)
   b. One to allow user to change / add a new one (Maintenance)

NOTE: A newer alternative to creating UI Maps would be to use UI Hints directly within the Business Object.

3. Create an info plug-in for the aggregator business object that concatenates together its dimension types and values.
4. Create a "Find Constituent Measuring Components" algorithm and plug it on the aggregator business object. This will be passed the aggregator measuring component and the from and to date/times. It will insert the constituent measuring component IDs and the respective from / to date-time of each onto a temporary table.
5. Create a measuring component type instance and reference the new aggregator measuring component business object (as well as the types of constituent measuring component types that should be aggregated).
6. Create a query zone for Consumption Statistics search to allow users to find the aggregator measuring component.

Optional steps:

1. Create a business object for the dimension scanner activity. This should be a child business object of the base package dimension scanner business object.
2. Create UI maps for the activity business object, as follows:
   a. One to display the dimension scanner activity (Display).
   b. One to allow users to change/add a new one (Maintenance).

   NOTE: A newer alternative to creating UI Maps would be to use UI Hints directly within the Business Object.

3. Create an info plug-in that will describe what it scans.
4. Create an Enter algorithm on the Scan state that finds distinct combinations of the dimensional values and creates new aggregator measuring components when new ones are detected.

NOTE: You can reuse the base package deferred monitor **Aggregation Dimension Scanner Monitor** (D1-ADS) batch control .

Dynamic Aggregation

Dynamic Aggregation Configuration Overview

Standard aggregation processing uses specific algorithms configured to work with a specified set of dimensions. For example, the base package Service Type and Postal aggregation uses specific algorithms for dimension scanning and finding constituent measuring components.

In contrast, dynamic aggregation uses dynamic queries for the dimension scanning, find constituents, and aggregation processes. These dynamic queries are based on configuration of administrative data, including:

- **Data Sources** define the source of data to be aggregated, such as measurement data from usage subscriptions linked to a service point, badged or unbadged items, or other sources of data.
- **Aggregation Measuring Component Types** define the most important properties of aggregation measuring components used to store aggregated data.
- **Aggregation Groups** define the ordering of a series of related aggregations and the schedule of aggregation.
- **Measuring Component Sets** define the dimensions and criteria by which aggregation will be performed.

Aggregation Master Configuration

The Aggregation Master Configuration defines common parameters used with aggregation processing, including:

- Batch Controls used by the dimension scan and aggregation process, for both scheduled and adhoc execution.
- Details for how the system compares interval and billing data for the same set of customers in order to use only one source or other.

Data Sources

Understanding Data Sources

Data Sources define the source of data to be aggregated, such as measurement data from usage subscriptions linked to a service point, badged or unbadged items, or measuring component sets. Data sources also define configurations used when generating dynamic queries used by aggregation processing.

Data Source Classes

Data sources are defined by classes, each of which is based on a specific business object, and determines the type of configuration used by aggregation processing. More specifically, the data types configured on the data source are used when generating dynamic queries used by aggregation processing.

The table below lists the data source classes and the business object and configuration options for each. Consult the embedded help for more information about the specific configuration options used by each data source class.

Data Source Class	Usage	Business Object	Configuration Options
Measurements from US Service Points	Used when aggregating data from measuring components linked to service points via a usage subscription.	D1-DataSourceMeasurements	Measuring Component Types
Badged Items	Used when aggregating data from badged items.	D1-BadgedItemsDataSource	Item Types
Billed Service Quantities	Used when aggregating data based on billed services quantities.	D1-DataSourceBilledSerQuantity	Usage Subscription Types
Measuring Component Sets	Used when aggregating data from measuring component sets. Data sources of this class are used with composite aggregation.	D1-DataSourceMCSet	Usage Subscription Types
Unbadged Items	Used when aggregating data from unbadged items.	D1-DataSourceUnbadgedItems	Item Types
Measurements from US Direct Links	Used when aggregating data from measuring components directly linked to a usage subscription.	D1-DataSourceMeasurementDirect	Measuring Component Types
Service Point Quantities	Used when aggregating data based on service point quantities.	D1-DataSourceSPQuantities	Service Point Quantity Types
Attribute and Measurement Data Snapshot	Used when aggregating data based on data snapshots	D1-MasterAndMsrmtDataSnapshot	Measurement Data Snapshot Types

Data Source Template SQL

Data sources also provide template structured query language (SQL) used when generating dynamic queries used by aggregation processing. This template SQL is used as the basis for dynamic queries, and is extended by additional SQL generated from the data source configuration as well as configuration defined for measuring component sets, measuring components, and measuring component types that reference the data source. Template SQL is provided via the Data Source SQL (D1-DataSourceSQLLookup) extendable lookup. Values for this extendable lookup are used as the **Option Value** for the following business object options on the data source business object.

- **Dimension Scanner SQL**: The template SQL used when dimension scanning. This SQL is extended by the criteria, dimensions, and individually managed items configured on the measuring component sets being processed.
- **Find Constituents SQL**: The template SQL used when finding constituents. This SQL is extended by attributes defined on the aggregator measuring components being processed.
- **Aggregation SQL**: The template SQL used when performing aggregation. This SQL is extended by the value identifiers defined on the measuring component type of the aggregator measuring components being processed.

Configuring Data Sources

This portal is used to display and maintain data sources.

Refer to Understanding Data Sources for more information.

You can access the portal by selecting **Admin**, then **Aggregation**, then **Data Source**.

The following zones may appear as part of the portal's **Main** tab page:

- **Data Source List**: This zone lists all data source records. Broadcast a record to display the details of the selected record.
- **Aggregation Data Source**: This zone provides information about the selected data source.

Configuring Data Source Template SQL

Template SQL used when generating dynamic queries used in aggregation processing is defined in the Data Source SQL (D1-DataSourceSQLLookup) extendable lookup. The base package data source business objects reference appropriate base package values from this extendable lookup.

For example, the **Aggregation SQL** option on the Measurements from US Service Points data source business object (D1-DataSourceMeasurements) references the “D1_MSRMT_FROM_US_SP_MCTYPE_SQL” value from the Data Source SQL extendable lookup.

If an implementation requires template SQL that differs from the base package, custom extendable lookup values can be created and referenced on the data source business object.

When creating custom template SQL, start from an existing extendable lookup value. Select the value that most closely matches your requirements and use the **Duplicate** function to create a copy, and edit the SQL as needed.

Use the following procedure to add a custom extendable lookup value to a data source business object.

1. Add a business object option of the appropriate type to the data source business object. For instance, if creating custom SQL for aggregation, add an **Aggregation SQL** option.
2. Enter a **Sequence** higher than the base package option.
3. Specify the custom extendable lookup value in the **Option Value** field.

Dynamic Aggregation Measuring Component Types

Understanding Dynamic Aggregation Measuring Component Types

Like other measuring component types, dynamic aggregation measuring component types define the most important properties of a measuring component. In this case, they define the properties of aggregator measuring components.

Measuring component types used by dynamic aggregation use many of the same attributes as other types of measuring components, but some attributes are used differently. Some attributes used specifically by dynamic aggregation include:

Measuring Component Business Object: This should be an aggregator measuring component business object, such as one of the following:

- Aggregator Interval (D1-AggregatorInterval)
- Aggregator Scalar (D1-AggregatorScalar)
- Aggregator Customer Quantities (D1-AggregatorCustomerQuantity)
- Aggregator Items Measuring Component (D1-AggregatorItems)

Measurement Business Object: Aggregated measurements can be based on either the Measurement (D1-Measurement) business object or the Aggregation Measurement (D1-AggregationMeasurement) business object (based on the Aggregation Measurement maintenance object).

Data Source: The source of the data that will be aggregated and the specific type of data that should be aggregated. Note that the **Service Type** of the data source, as determined by its associated objects (e.g. Measuring Component Type, Device Type, Service Point Type), must match the **Service Type** for this measuring component type.

Value Identifiers: These store the values of UOM, TOU, and SQI that identify the measurements to be aggregated. For each value identifier either the **Value Calculation Method** or **Value Derivation Algorithm** must be defined (not both). These define how measurement values are calculated or derived during aggregation processing.

Refer to Understanding Measuring Component Types for more information about measuring component types.

Configuring Dynamic Aggregation Measuring Component Types

The **Measuring Component Type** portal is used to display and maintain dynamic aggregation measuring components.

Refer to Understanding Dynamic Aggregation Measuring Component Types for more information about dynamic aggregation measuring components.

You can access the **Measuring Component Type** portal from **Admin > Device > Measuring Component**.

Refer to Configuring Measuring Component Types for more information about working with measuring component types.

Aggregation Groups

Understanding Aggregation Groups

Aggregation Groups define the ordering of a series of related aggregations based on a set of configured Measuring Component Sets. Aggregation groups define the following attributes for aggregations performed for its Measuring Component Sets:

- The **Time Zone** for aggregated data
- **Aggregation Scheduling** can be based on either a Defined Lag, or a Processing Timetable.
  - **Defined Lag** scheduling uses the following parameters:

- **Aggregation Cut Off Time**: The end time for aggregation calculations performed for Aggregation Measuring Components for Measuring Component Sets associated with the Aggregation Group. This is used to ensure a consistent end time for aggregation periods. This is especially useful when aggregating other aggregations.
- **Aggregation Lag**: The number of days between the date on which aggregation calculations are performed and the end date of the aggregation period. This defines the time period between the aggregation calculation date and the aggregation horizon that serves to allow all measurements to arrive. This together with the Aggregation Horizon is used to determine the start and end dates of an aggregation period. For example, with an Aggregation Horizon of 5 and an Aggregation Lag of 2, aggregation calculations performed on January 9 would be for an aggregation period of January 3 through January 7. The next day (January 10), the aggregation period would shift to January 4 through January 8.
- **Aggregation Horizon**: The number of days in the aggregation period for Aggregation Measuring Components for Measuring Component Sets associated with the Aggregation Grou. This reflects the time period during which there's a potential change in measurement data for one or more of the measuring components associated with Aggregation Measuring Components. This together with the Aggregation Lag is used to determine the start and end dates of an aggregation period.

- **Processing Timetable** scheduling uses a Processing Timetable Type to drive aggregation schedule.

Aggregation groups can be used to perform aggregations for Analytical or Settlement purposes (defined by the Aggregation Category).

Aggregation Group Runs

Aggregation group runs represent individual aggregation executions for an individual aggregation group. Aggregation group runs are used to capture the relevant details for an aggregation and track the status of the execution of the aggregation for each of the Measuring Component Sets for the aggregation group.

Aggregation group runs for a specific aggregation group are listed in the **Aggregation Group Run** zone on the **Aggregation Group** portal. This zone displays the following for each aggregation group run:

- Run Number
- Run Type (Scheduled vs. Ad Hoc)
- Status (Pending, In Progress, Completed, or Error)
- Aggregation Horizon Start
- Aggregation Horizon End
- Details

Details for a specific aggregation group run can be viewed by broadcasting the run in the **Aggregation Group Run** zone. The following is displayed for the selected run in the **Aggregation Group Run Details** zone:

- Sequence
- Measuring Component Set
- Initiation Method
- Batch Run

Configuring Aggregation Groups

This portal is used to display and maintain aggregation groups.

Refer to Understanding Aggregation Groups for more information.

You can access the portal by selecting **Admin**, then **Aggregation**, **Aggregation Group**.

The following zones may appear as part of the portal's **Main** tab page:

- **Aggregation Group List**: This zone lists all aggregation group records. Broadcast a record to display the details of the selected record.
- **Aggregation Group**: This zone provides information about a selected aggregation group.
- **Aggregation Measuring Component Sets**: This zone lists measuring component sets associated with the selected aggregation group. Click **Resequence** to change the order of the measuring component sets in the aggregation group. Click **Add** to add a new measuring component set to the aggregation group.
- **Aggregation Group Run**: This zone lists individual aggregation runs performed for the selected aggregation group. Broadcast a record to display the details of the selected record. This zone displays up to 10 records. If there are more than 10 records, use the **Previous** and **Next** buttons to navigate between pages of records. Click **Add** to execute an ad hoc aggregation group run.
- **Aggregation Group Run Details**: This zone displays details for a selected aggregation group run.

Measuring Component Sets

Understanding Measuring Component Sets

Measuring component sets are used to define the dimensions and criteria by which aggregation will be performed.

A measuring component set's **Measuring Component Type** defines the Measuring Component and Measurement Type to be used in capturing aggregation results, as well as the aggregation functions that will be used on the source transnational data and the Data Source that will identify where that transnational data will be retrieved from.

Measuring component sets are associated with an **Aggregation Group**, which controls when and how aggregation is processed. This includes the sequence in which the aggregation group's measuring component sets are processed, as well as the method by which aggregation processing is initiated for each (this information can be found in the **General Processing** section).

A measuring component set's **Measuring Component Set Class** defines the way in which the measuring component set is used in the aggregation process. Measuring component set classes are based on the measuring component set's business object, and can include:

- **Aggregation Foundation** measuring component sets are used to aggregate transactional data, such as measurements from service points linked via usage subscriptions or billed service quantities.
- **Aggregation Composite** measuring component sets are used to aggregate data from a set of other aggregations (which can be based on either foundation or other composite measuring component sets).
- **Adhoc** measuring component sets are used to aggregate data based on an adhoc, user-defined set of measuring components.

See **Foundation and Composite Aggregation** in Dynamic Aggregation Processing for more information.

Dimensions, Criteria, and Individually Managed Items

The **Dimensions** and **Criteria** defined for the measuring component set are used by the dimension scanning process to identify individual aggregation measuring components that should be created. Measuring components are also created for each **Individually Managed Item** for which data matches the item's configuration.

- **Dimensions** identify the attributes by which data should be aggregated. Dimensions can include characteristics, identifiers, market participants, and values from specific tables and columns. For example, aggregating data by loss profile code, service point type and service provider would require defining dimensions for a Loss Profile Code characteristic, the Service Point Type column from the Service Point table, and the Service Provider column from the Device table.

NOTE: When configuring measuring component sets for aggregation of data from measurement data snapshots, the **Source Type** and **Source Entity** should be “Attribute Data Snapshot”.

- **Criteria** define the ways in which the dimensions are evaluated through a series of inclusions, exclusions, or both. For example, to aggregate data for the “AGG-E-COM” and “AGG-E-RES” service point types, but NOT for devices from the “ELEC” service provider, criteria could be defined to include both of service point type values and exclude the “ELEC” service provider.

  NOTE: When configuring measuring component sets for aggregation of data from measurement data snapshots, the **Source Type** and **Source Entity** should be “Attribute Data Snapshot”.

- The criteria provided for **Individually Managed Items** is used to identify specific customers that should be aggregated in isolation from all other data that matches the **Dimensions** and **Criteria** defined for the measuring component set.

More specifically, the dimension scanning template SQL (derived from the data source) is extended by the **Dimensions**, **Criteria**, and **Individually Managed Items** configured on the measuring component set being processed.

Consult the embedded help for more information about the specific configuration options used by **Dimensions**, **Criteria**, and **Individually Managed Items**. The Source Type and Source Entity values are defined in the Aggregation Criteria Source Type (D1-CriteriaSourceTypeLookup) extendable lookup. Refer to Configuring Aggregation Criteria Source Types for more information about this extendable lookup.

The manner in which the criteria provided will be evaluated is defined in formulas configured in the **Criteria Processing** section. Consult the embedded help for more information about defining criteria processing formulas.

Configuring Measuring Component Sets

This portal is used to display and maintain measuring component sets.

Refer to Understanding Measuring Component Sets for more information.

You can access the portal by selecting **Admin**, then **Aggregation**, **Measuring Component Set**.

The following zones may appear as part of the portal's **Main** tab page:

- **Measuring Component Set**: This zone displays details of a selected measuring component set.
- **Dimensions**: This zone lists the dimensions used to identify the attributes by which the aggregation should be analyzed. Note: Data in this zone is edited by clicking the **Edit** button in the **Measuring Component Zone**.
- **Criteria**: This zone lists the criteria that limits the data to be aggregated through a series of inclusions, exclusions, or both. Note: Data in this zone is edited by clicking the **Edit** button in the **Measuring Component Zone**.
- **Individually Managed Items**: This zone lists the criteria used to identify specific customers that should be aggregated in isolation from all other data that matches the dimensions and criteria defined for the measuring component set. Note: Data in this zone is edited by clicking the **Edit** button in the **Measuring Component Zone**.
- **Measuring Component Set Participants**: This zone displays the measuring components that are in the Measuring Component Set Participant table for the measuring component set. For each measuring component, this zone displays the most recent and last measurement date/times.

Configuring Aggregation Criteria Source Types

The options in the **Source Type** and **Source Entity** drop-down lists used when configuring Dimensions, Criteria, and Individually Managed Items are defined in the Aggregation Criteria Source Type (D1-CriteriaSourceTypeLookup) extendable lookup. The measuring component set business objects reference this extendable lookup.

Each of the base package Source Type values (Characteristic, Identifier, Market Participant, and Table) are records in this extendable lookup. The Source Entity values for each of these are defined in the corresponding record.

For example, the “Table” Source Type defines the base package “Table” Source Entity values (Device, Device Event, Measuring Component, Service Point, Usage Subscription, and Usage), along with criteria path details for each (based on data source class).

If an implementation requires Source Type or Source Entity values that differ from the base package, custom extendable lookup values can be created.

When for creating custom values for this extendable lookup, start from an existing extendable lookup value. Select the value that most closely matches your requirements and use the **Duplicate** function to create a copy, and edit as needed.

Configuration Step-by-Step

This section outlines the steps involved when configuring dynamic aggregation.

Create Device and Service Point Administrative Data

Many of the objects used in dynamic aggregation reference other device- and service point-related administration data, such as measuring component types, service point types, usage subscription types, etc. Make sure this data is in place before configuring entities used with dynamic aggregation.

Create Data Sources

Create data sources for each source of data from which data will be aggregated. Note that implementations can have multiple data sources of each class with different data type configuration (Measuring Component Types, Item Types, Usage Subscription Types, etc.). For example, an implementation with 15 measuring component types might want to aggregate data differently based on subsets of measuring component types. In this case, each subset of measuring component types could be defined on a different data source.

If applicable, create custom Template SQL extendable lookup values and add them to Data Source business objects as appropriate.

Create Aggregation Measuring Component Types

Create the aggregation measuring component types that will be used to define the attributes of the measuring components that will be created during aggregation processing.

Aggregation measuring component types reference an appropriate Data Source. Continuing the example above, each subset of measuring component types (each defined on a separate data source) would require a separate aggregation measuring component type.

Create Aggregation Groups

Create the aggregation groups that will be used to control aggregation processing, including aggregation groups for both foundation and composite aggregation processing.

Note: When initially creating aggregation groups, define only the basic data for each (name, category, and optional time zone). Add measuring component sets to aggregation groups when creating the latter in the next step.

Create Measuring Component Sets

Create the measuring component sets that will define the “buckets” by which the aggregated data will be grouped.

Measuring component sets reference aggregation measuring component types and aggregation groups.

Create custom Source Type records (if applicable).

Revise Aggregation Groups (if needed)

After creating measuring component sets, adjust the sequence of each within its aggregation group as appropriate. For example, this would be necessary if measuring component sets are created and added to aggregation groups in an order that differs from their intended sequence.

Extending Dynamic Aggregation

The dynamic aggregation functionality provides extensive flexibility through configuration of data sources, measuring component types, and measuring component sets to handle complex aggregation requirements. However, some implementations may require functionality and configuration options beyond that provided in the base package. For instance, an implementation may need to aggregate data of a source and type not covered by the base package data sources or use different dimensions and criteria or queries for aggregation processing.

In circumstances like these, the dynamic aggregation functionality can be extended through use of the Oracle Utilities Application Framework's Configuration Tools (see Configuration Tools) and other options. This section outlines some of the possible approaches to extending this functionality.

Data Source Business Objects: If an implementation requires aggregation of a data type of source beyond those provided with the base package data source business objects, custom business objects can be created to support this requirement. Key considerations when creating custom data source business objects include:

- Configuration options such as "Sourced Measuring Component Types" or "Sourced Usage Subscription Types"
- Values for the "Aggregation SQL", "Dimension Scanner SQL", and "Find Constituents SQL" business object options from the Data Source SQL extendable lookup.
- Algorithms for the "Generate SQL For Data Source" system event that leverage the configuration options defined for the data source

Data Source SQL Extendable Lookup Values: If an implementation requires template SQL that differs from the base package, custom values for the Data Source SQL extendable lookup can be created and referenced on the data source business object. When creating custom template SQL, start from an existing extendable lookup value. Select the value that most closely matches the requirements and use the **Duplicate** function to create a copy, and edit the SQL as needed.

Aggregation Measuring Component Type Business Objects: If an implementation requires capturing data for measuring component types not provided in the base package, the base package aggregation measuring component type business objects can be extended as appropriate. Key considerations when creating custom measuring component type business objects include:

- Algorithms for the "Generate SQL For MC Type" system event that leverages the Value Identifiers and other information defined for the measuring component type

Aggregation Measuring Component Business Objects: If an implementation requires capturing data for measuring components not provided in the base package, the base package aggregation measuring component business objects can be extended as appropriate. Key considerations when creating custom measuring component business objects include:

- Algorithms for the "Generate SQL For Aggregator MC" system event that leverages the Value Identifiers and other information defined for the measuring component type
- Algorithms for the "Find Constituents" system event that uses the "Generate SQL for Aggregator MC" algorithm, data source configuration, and other configuration to find constituent data for aggregation
- Algorithms for the business object's "Active" state, including data aggregation and value derivation

Measuring Component Set Business Objects: If an implementation requires capturing data for measuring component sets not provided in the base package, the base package aggregation measuring component set business objects can be extended as appropriate. Key considerations when creating custom measuring component set business objects include:

- Algorithms for the "Generate SQL For MC Set" system event

- Algorithms for the business object's lifecycle states, as appropriate

Aggregation Criteria Source Type Extendable Lookup Values: If an implementation requires Source Type or Source Entity values for measuring component sets that differ from the base package, custom extendable lookup values can be created. When creating custom values for this extendable lookup, start from an existing extendable lookup value. Select the value that most closely matches the requirements and use the **Duplicate** function to create a copy, and edit as needed.

Consumption Synchronization

Configuring Consumption Synchronization

> **NOTE:** Refer to Introduction to Consumption Sync for additional functional information about how consumption synchronization works.

Keeping consumption synchronized between two measuring components that meter the same quantity but at different frequencies is a complicated task. As such the configuration for this process is diffuse and requires settings across several key areas of the system to be aligned. This section will help guide you through the process of configuring consumption synchronization.

The consumption synchronization process is really a collection of processes within Oracle Utilities Meter Data Management that all work together to ensure that quantities between related channels remain consistent:

- Estimation VEE Rules: several rules align consumption between related channels. These rules allow estimations to be refined based on higher quality measurements from a related channel. These rules are core to the consumption synchronization process. In simple scenarios where a few intervals are missing for an interval measuring component these rules are all that are necessary for synchronization with the related channel (note: it requires that the related channels data for that same time period has already been processed through to final measurements). For more information on these rules please visit the following sections:
  - Interval Adjustment from Scalar
  - Scalar Calculation from Interval
  - Scalar Proration
  - Sum Check
  - Final Measurement Validation

  > **NOTE:** Refer to About IMD Estimations for more information about these rules

- Periodic Estimation: both the interval and scalar variations of periodic estimation play an important role in consumption synchronization by ensuring that both channels of data are without gaps. In simple scenarios where one channel is missing data and the other is not periodic estimation is all that is require to produce synchronized consumption. It is important to note that this process itself does not perform estimations but rather it is responsible for creating estimation initial measurements to trigger the estimation VEE rules.
- Consumption Synchronization Activities: these activities work to fix alignment issues that occur when data for the related channels are processed out of order or when there are complex outage scenarios. These activities are created when higher quality data is received for one channel and the related channel has measurements that are eligible to be recalculated and adjusted to align the total consumption for the period across the two channels. It is important to note that this process itself does not perform estimations but rather it is responsible for creating estimation initial measurements to trigger the estimation VEE rules.

Device Configuration Type

As a default the system will not generate consumption synch activities for related measuring components. Turning on consumption synchronization activities is done through a few key fields on the device configuration type:

- **Keep Consumption Reference MC in Sync**: defines if the related MCs should be kept in sync. It can be configured to provide a one way synchronization from primary to secondary or a two way synchronization between primary and secondary.
- **Minimum Condition to Sync Primary MC**: when the secondary measuring component can initiate synchronization of the primary this provides an ability to limit those situations to when the incoming data is of a minimum quality.
- **Sum Check VEE Exception Type**: provides further ability to limit initiation of synchronization. When configured the synchronization activities will only be created when the initial measurement being processed encounters a VEE exception of the type configured. More specifically, a sum check VEE exception which will indicate that the two channels are out of synch by a minimum tolerance amount. This can be used to avoid synchronization either when there is no difference between the channels or when there is only a small difference between the channels.

NOTE: Additional detail about these fields can be found in the embedded help for the device configuration type.

Register Auto-Read Measuring Component Type

The configuration available for the register auto-read measuring component type has an indirect impact on the consumption synchronization process. This configuration is primarily intended to allow for a register to re-evaluate previously created estimations when new more accurate readings are received even when no related measuring components exist. Where this has impact to the consumption synchronization process is that when a new scalar reading is received after a period of estimation it will result in the time period for that new initial measurement being expanded into the past. This is because any estimates prior to this new higher quality initial measurement will be logically removed and the start reading for the initial measurement will in turn be the last non-estimated measurement prior to the initial measurement. This creates an initial measurement that spans the entire period for which estimations exist and as a result when a consumption synchronization activity is generated it will result in that same period of time being re-evaluated on the related measuring component.

This functionality is controlled by a few key fields on the register auto-read measuring component type:

- **Ignore Estimates as IMD Start Reading**: controls whether estimates directly previous to newly received incoming initial measurement should be logically removed when that newly received initial measurement data is non-estimated.
- **Flag Future Estimates as Do Not Use**: controls whether estimates that come directly after newly received incoming initial measurements should be logically removed when that newly received initial measurement data is non-estimated
- **Actuals or Corrections Initial Re-Estimation:** works in tandem with the above two fields. When either scenario results in measurements being logically removed this field, when turned on, will reset the last contiguous measurement date/time for the measuring component making it eligible once more for periodic estimation. With the end result being that those estimated measurements that were removed would recreated as new estimates.

Measuring Components

In order for consumption synchronization to work it must know that two measuring components are related in a way that indicates they are measuring the same consumption. This is achieved by configuring the "Consumption Reference Measuring Component" field on the measuring component.

Not only does this establish the relationship but it also establishes which measuring component is considered to be primary and which is considered to be secondary.

A measuring component is considered to be secondary when it holds the relationship to the Consumption Check Measuring Component (the primary measuring component).

For example, consider the following related measuring components.

- Scalar Measuring Component: ER-SM-007 / 1 / Electric kWh Daily

- Interval Measuring Component: ER-SM-007 / 2 / Electric kWh 60min

If the scalar measuring component is the primary measuring component, it does NOT specify a "Consumption Reference Measuring Component", and the interval measuring component specifies the scalar measuring component as the "Consumption Reference Measuring Component".

If the interval measuring component is the primary measuring component, it does NOT specify a "Consumption Reference Measuring Component", and the scalar measuring component specifies the interval measuring component as the "Consumption Reference Measuring Component".

It is important to note that pending initial measurements for the secondary measuring component are processed by the D1-IMD batch process after the initial measurements for the primary measuring component. This ensures that the primary measuring component's final measurements will be available to the secondary measuring component's initial measurements VEE process to provide better quality estimations and validations.

Initial Load and Manual Initial Measurement Algorithms

Key to the consumption synchronization process are the algorithms that reside on initial load and manual initial measurements for both scalar and interval measuring components. These algorithms contain logic to identify final measurements on a related channel that are consumption synchronization eligible (typically estimated).

These algorithms allow definition of:

- The consumption synchronization activity created
- The condition range that defines consumption synchronization eligible final measurements

Please refer to the algorithm type descriptions for more information:

Algorithm Type	Description	Where used
D1-UPD-DTMC	Update Latest Measurement Date/Time on MC with Consumption Sync	D1-InitialLoadIMDInterval D1-ManualIMDInterval
D1-UDTSCMCRE	Update Latest Measurement Date/Time on Scalar MC with Consumption Sync	D1-InitialLoadIMDScalar D1-ManualIMDScalar

Consumption Synchronization Activities

The consumption synchronization activities are initiated by initial measurements and are responsible for generating the appropriate estimation initial measurements to re-evaluate any consumption synchronization eligible final measurements for the related measuring component being processed.

These activities can be associated to one or more initiating initial measurements and are able to handle a broad combination of time periods which are not required to be contiguous.

If any generated estimation initial measurement does not finalize processing will be halted and details about the failed estimation initial measurement will be provided.

The following table identifies the catalogue of consumption synchronization activities:

Activity Type	Description	Activity Business Object
Related MC Consumption Sync - Scalar	This activity is generated to re-evaluate consumption sync eligible final measurements on scalar measuring components. One estimation initial measurement will be created per measurement needing to be re-evaluated.	D1-RelMCREScalar
Related MC Consumption Sync - Interval	This activity is generated to re-evaluate consumption sync eligible final measurements	D1-RelMCREInterval

	on interval measuring components. One estimation initial measurement will be created per contiguous set of interval measurements needing to be re-evaluated. Note the generated initial measurements will include non-consumption sync eligible final measurements which will help to feed into the estimation process to provide more accurate results.	
Gap Period Consumption Sync	This activity will re-evaluate both measuring components in the relationship for a wider time period than the initiating initial measurement. Specifically the time period will be the total contiguous period of consumption sync eligible final measurements across both channels in the relationship.	D1-GapPeriodConsumpnSync

Each consumption synchronization activity has an algorithm that performs the core logic of the consumption synchronization process.

These algorithms allow definition of:

- The condition range that defines consumption synchronization eligible final measurements

Please refer to the algorithm type descriptions for more information:

Algorithm Type	Description	Where used
D1-RE-SCMC	Scalar MC Consumption Sync	D1-RelMCREScalar
D1-RE-INTVMC	Interval MC Consumption Sync	D1-RelMCREInterval
D1-GAPCSYNC	Gap Period Consumption Sync	D1-GapPeriodConsumpnSync

Periodic Estimation

The algorithm that initiates periodic estimation which is plugged in to the Smart Meter device business object has a key configuration setting related to consumption synchronization:

- **MC Type to Process First**: determines whether interval or scalar measuring components should be estimated first. This should be set to process whichever measuring component type is considered to be primary first.

Please refer to the algorithm type description for more information:

Algorithm Type	Description	Where used
D1-PERESTM	Periodic Estimation	D1-SmartMeter

Related Batch Controls

There are a few batches involved with Consumption Synchronization:

- **Related MC Consumption Sync** (D1-RMCRE): this processes any pending consumption synchronization activities.
- **Related MC Consumption Sync - Retry Act** (D1-RMCRR): this retries any consumption synchronization activities that fail to the Issue Detected state.

Dashboards

Configuring the MDM Operational Dashboard

This section describes the process for configuring the MDM Operational Dashboard.

Refer to the Using the MDM Operational Dashboard section of the *Oracle Utilities Meter Solution Business User Guide* for a information on this functionality.

Configuration for the MDM Operational Dashboard is performed by adding or changing a Master Configuration. You can access the portal from the **Admin > General > Master Configuration**.

Once the Master Configuration search screen returns, configure both the **MDM Operational Dashboard Configuration** record. Use the Add button beside the record to configure for the first time. If a record has already been added, then click the Edit button instead. Use the embedded help to guide you through the meaning of each configuration field.

This dashboard also leverages a method of pre-staging data known as Statistics Snapshots. Refer to the *Framework Administrative User Guide* for more information on Statistics Snapshots. For additional information see About Statistics in the Oracle Utilities Application Framework *Administrative User Guide*.

A method for tracking Performance Targets (also known as Service Level Agreements) is available within the MDM Operational Dashboard as well. To understand how to set up Performance Targets, refer to About Performance Targets in the Oracle Utilities Application Framework *Administrative User Guide*. The Performance Targets you configure will then be leveraged on the **Batch Performance** and/or **Database** tabs of the MDM Operational Dashboard.

Configuring the Service Order Operational Dashboard

This section describes the process for configuring the Service Order Operational Dashboard.

Refer to the Using the Service Order Operational Dashboard section of the *Oracle Utilities Meter Solution Business User Guide* for more functional information.

Configuration for the Service Order Operational Dashboard is performed by adding or changing a Master Configuration. You can access the portal from the **Admin > General > Master Configuration**.

Once the Master Configuration search screen returns, locate the **Service Order Management Master Configuration** record. Use the Add button beside the record to configure for the first time. If a record has already been added, then click the Edit button instead. The **Chart Options** section is the primary area for adding configuration that will affect the dashboard. Use the embedded help to guide you through the meaning of each configuration field.

Configuring the Service Order Trends Dashboard

This section describes the process for configuring the Service Order Trends Dashboard.

Refer to the Using the Service Order Trends Dashboard section of the *Oracle Utilities Meter Solution Business User Guide* for more functional information.

Configuration for the Service Order Trends Dashboard is performed by adding or changing a Master Configuration. You can access the portal from the **Admin > General > Master Configuration**.

Once the Master Configuration search screen returns, locate the **Service Order Management Master Configuration** record. Use the Add button beside the record to configure for the first time. If a record has already been added, then

click the Edit button instead. The **Chart Options** section is the primary area for adding configuration that will affect the dashboard. Use the embedded help to guide you through the meaning of each configuration field.

Data Extracts

This section describes functionality that support extract and export of data for use in external applications and systems.

Configurable Consumption Extracts

Configurable Consumption Extracts are measurement data extracts for a set of usage subscriptions optionally related to a specific market participant.

Configuration options for these extracts include;

- Type of Market Participant, based on Service Point Market Participant or Usage Subscription Market Participant tables and market relationship
- Usage Subscriptions Types associated with the market participant (typically a customer information system receiving bill determinants)
- Unique combinations of unit of measure (UOM), time of use (TOU), and service quantity identifier (SQI) associated with the usage subscriptions
- Specific identifiers used to identify the usage subscriptions, service points, and devices associated with extracted measurement data
- Date and time parameters and frequency of extract
- Extract file naming and compression options

Consumption extracts are exported to a file in a specified file location, either on an application server (in an on-premises implementation) or an Oracle Cloud Infrastructure Object Storage location.

Consumption extracts are created via Consumption Extract Requests. See Consumption Extract Requests in the *Business User Guide*.

This section describes the configuration needed for these types of consumption extracts.

Consumption Extract Types

Configuration for consumption extracts are defined via a Consumption Extract Type. See Consumption Extract Type in the *Administrative User Guide* for more information about creating these records.

Configurable consumption extract types should be created using the Extract US Consumption to File (D1-ExtractUSConsumptionToFile) business object. Refer to the embedded help on this business object for more details about the parameters defined for consumption extract types.

Extract US Consumption to File consumption extract types define the following parameters:

- The market participants for which measurement data will be extracted. You define market participants based on their type (Service Point Market Participant or Usage Subscription Market Participant) and market relationship type (retailer, distributor, supplier, and so on), and a specific market participant. Note: If Market Participant Specific is set to "No", the market participant used is based on the Usage Recipient on the Usage Subscription or its Usage Subscription Type.
- The consumption type (interval or scalar)
- One or more Usage Subscription Types and a related unique UOM/TOU/SQI combination
- The specific ID types to include in the extract file for the Usage Subscription, Service Point, child Service Point (if applicable), and Device

- Execution parameters, including:
  - The date and time through which data has been extracted for this consumption extract type
  - A Request Type. Request types used with consumption requests should be created using the Consumption Extract Request Type (D2-IntervalDataExtRepoType) business object, and should reference the Consumption Extract (D2-IntervalDataExtRepository) business object in the **Request Business Object** field. (see The Big Picture of Requests in the *Application Framework Administrative User Guide*)
  - The frequency of extracts for this type (adhoc or daily)
  - Batch controls used for the extract process (see Batch Controls below)
- File parameters, including:
  - The file location where extract files will be saved (see External File Storage in the *Application Framework Administrative User Guide*)
  - An optional child folder in the file location
  - File naming options (including an optional prefix and date/time stamp)
  - Whether or not to compress the extract file (file compression is performed using gzip)

Batch Controls

The data extract process and creation of export files is performed by the following batch controls:

- **Extract Current US Intervals to File** (D1-ECUSI): Used for daily extracts of interval consumption.
- **Extract Replacement US Intervals to File** (D1-ERUSI): Used for extracts of replacement historical interval consumption. This process runs against Change Data Capture records in the General Process table (see **Initial Measurement Subsequent Correction Change Data Capture** below).
- **Extract Current US Scalar Reads to File** (D1-ECUSS): Used for daily extracts of scalar consumption.
- **Extract Replacement US Scalar Reads to File** (D1-ERUSI): Used for extracts of replacement scalar interval consumption. This process runs against Change Data Capture records in the General Process table (see **Initial Measurement Subsequent Correction Change Data Capture** below).

These batch controls are run for a specific Consumption Extract Type and date range. Batch processing is threaded by on Usage Subscription ID.

Initial Measurement Subsequent Correction Change Data Capture

The follow algorithms maintain measuring component date/times based on the latest measurement date time for the measuring component. These algorithms also check for a Consumption Extract Type that applies to the current initial measurement's measuring component and identifies whether the initial measurement is for a date/time that has already been extracted.

Algorithm	Description	Consumption Type
D1-UPD-DTMC	Update Latest Measurement Date/Time on MC with Consumption Sync	Interval
D1-UDTSCMCRE	Update Latest Measurement Date/Time on Scalar MC with Consumption Sync	Scalar
D1-UPD-DTMCF	Update Latest Date/Time on MC from a Future Measurement	Interval
D1-UPDDTSCMC	Update Latest Date/Time on Scalar Measuring Component	Scalar

If an initial measurement is for a date/time that has already been extracted, these algorithms create Change Data Capture records in the General Process table that capture the batch code, batch run number, and the initial measurement's ID. The "Replacement" batch processes run off of these records.

Note that the following parameter must be set to “true” on the above algorithms to enable historical change data capture:

- Capture CET Historical Measurement Changes: true

File Format - Interval

The interval extract process produces JavaScript Object Notation (JSON) files with the extract data described in the following table.

Note that interval extract files can store up to a maximum of 300 intervals. This means that export files can store approximately:

- 12.5 days worth of hourly data
- 3 days worth of 15 minute data
- 1 day worth of 5 minute data

Field	Sample Value	Comments
usId	995647003500	Usage Subscription ID defined by the Consumption Extract Type
usType	E-RES	Usage Subscription type
pSpId	458081	Parent Service Point ID, Null when there is no parent service point.
spId	458081_001	Child Service Point ID This is the Service Point ID defined by the Consumption Extract Type
dvcId	SN823287793219	Device ID defined by the Consumption Extract Type
uomTouSqi	KWH//	UOM / TOU / SQI, separated by forward slashes ("/")
tz	US/Eastern	Time zone of the service point. This is the time zone name from the CI_ TIME_ZONE table not the time zone code.
intPerDay	288	Number of Intervals for the day, will be 1 hour less or 1 hour more of intervals for DST transitions
intSize	300	Length of each interval in seconds
mktPart	DIRECT_ENERGY	Market Participant ID of the market participant (if applicable)
stDttm	2020-01-01T00:00:00-05:00	Start of the day, similar to initial measurement, this represents the start of the first interval.
q1	1.325	Measurement Value for the first interval, the measurement date/time for this interval would be the stDttm plus the intSize
c1	301000	Measurement Condition for the first interval. Null when "Regular" (501000)

Field	Sample Value	Comments
q2	1.230	
c2	301000	
...	q/c 3 through 299 here	
q300	.9134	
c300	30100	

Example:

```
{"usId":"995647003500", "usType":"E-
RES", "pSpId":"458081", "spId":"458081_001", "dvcId":"SN823287793219", "uomTouSqi":"KWH//",
 "tz":"US/Eastern", "intPerDay":"288", "intSize":"300", "mktPart":"DIRECT_ENERGY",
 "stDttm":"2020-01-01T00:00:00-05:00", "q1":"1.325", "c1":"301000", "q2":"1.325",
 "c2":"301000" "q3":"1.325", "c3":"301000",
<intervening intervals>, "q300":"1.325", "c300":"301000"}
```

File Format - Scalar

The scalar extract process produces JavaScript Object Notation (JSON) files with the extract data described in the following table.

Name	Sample Value	Comments
c	301000	Measurement Condition for the scalar reading Null when "Regular" (501000)
dvcId	SN823287793219	Device ID defined by the Consumption Extract Type
enDttm	2020-01-02T00:00:00-05:00	End date/time of the reading
mktPart	DIRECT_ENERGY	Market Participant ID of the market participant (if applicable)
pSpId	458081	
q	1.325	Measurement Value for scalar reading
r	678645	Dial reading of the meter
spId	458081_001	Service Point ID defined by the Consumption Extract Type
stDttm	2020-01-01T00:00:00-05:00	Start date/time of the reading (the End Date/ Time of the previous reading)
tz	US/Eastern	Time zone of the service point. This is the time zone name from the CI_ TIME_ZONE table not the time zone code.
uomTouSqi	KWH//	UOM / TOU / SQI, separated by forward slashes ("/")
usId	995647003500	Usage Subscription ID defined by the Consumption Extract Type
usType	E-RES	Usage subscription type

Example (3 records):


```
{"usId":"995647003500", "usType":"E-
RES", "pSpId":"458081", "spId":"458081_001", "dvcId":"SN823287793219", "uomTouSqi":"KWH//",
 "tz":"US/Eastern", "mktPart":"DIRECT_ENERGY", "stDttm":"2020-01-01T00:00:00-05:00",
 "enDttm":"2020-01-02T00:00:00-05:00", "q":"1.325", "c":"301000", "r":"678645"}
{"usId":"995647003500", "usType":"E-
RES", "pSpId":"458081", "spId":"458081_001", "dvcId":"SN823287793219", "uomTouSqi":"KWH//",
 "tz":"US/Eastern", "mktPart":"DIRECT_ENERGY", "stDttm":"2020-01-01T00:00:00-05:00",
 "enDttm":"2020-01-02T00:00:00-05:00", "q":"1.325", "c":"301000", "r":"678645"}
{"usId":"995647003500", "usType":"E-
RES", "pSpId":"458081", "spId":"458081_001", "dvcId":"SN823287793219", "uomTouSqi":"KWH//",
 "tz":"US/Eastern", "mktPart":"DIRECT_ENERGY", "stDttm":"2020-01-01T00:00:00-05:00",
 "enDttm":"2020-01-02T00:00:00-05:00", "q":"1.325", "c":"301000", "r":"678645"}
{"usId":"995647003500", "usType":"E-
RES", "pSpId":"458081", "spId":"458081_001", "dvcId":"SN823287793219", "uomTouSqi":"KWH//",
 "tz":"US/Eastern", "mktPart":"DIRECT_ENERGY", "stDttm":"2020-01-01T00:00:00-05:00",
 "enDttm":"2020-01-02T00:00:00-05:00", "q":"1.325", "c":"301000", "r":"678645"}
```


Extending Extract Processing

Some implementations may wish to extend and/or customize the extract process based on specific requirements.

Extending List of IDs To Use

Consumption Extract Types define specific types of IDs to use for Usage Subscriptions, Service Points, child Service Points (if applicable), and Devices. If an implementation wishes to extend the types of IDs available, they need to add appropriate values to the lookup flags used by each of ID type. The table below lists the lookup codes used by the "IDs to Use" fields.

ID Type	Lookup Flag
Usage Subscription ID	Usage Subscription Identifier Type (US_ID_TYPE_FLG)
Service Point ID (used for both parent and child service points)	Service Point Identifier Type (SP_ID_TYPE_FLG)
Device ID	Device Identifier Type (DVC_ID_TYPE_FLG)

Processing Records and Creating Output Files

The extract process leverages the generic Plug-in Driven Background Processes functionality but works in a slightly different manner. Instead of using the Select Records algorithm to retrieve the IDs of the individual records to be processed, that algorithm returns a range of Usage Subscription IDs to be extracted and the Process Records algorithm performs the detailed query to extract the range of Usage Subscription IDs.

This provides two benefits:

1. There is no limitation on the length of the query in the Process Records algorithm.
2. Retrieving data for many Usage Subscriptions at a single time is significantly more efficient.

This means that any required customization can be achieved by creating a new custom (CM) batch control and duplicating the Process Records Groovy script. Changes can be made within the custom Groovy script to the extraction SQL to retrieve additional information or provide additional selection criteria. If any changes are needed to the file format that can be done within the record processing logic that populates the file.

> **NOTE:** The Process Records scripts include a number of steps each containing specific processing logic and SQL based on the configuration of the Market Participants on the Consumption Extract Type as well as whether or not the specified UOMs measure peak quantities or not. Be sure to edit each of these steps as appropriate when creating customized versions of these scripts.

Maintenance Object Data Export

This section describes the process by which data can be exported from maintenance objects used with Oracle Utilities Customer Cloud Service and Oracle Utilities Meter Solution Cloud Service for use in external applications and systems.

This section provides details regarding how the data export process is used with specific maintenance objects. See Data Export in the *Application Framework Administrative User Guide* for a general description of the data export process.

Enabling Maintenance Objects for Generalized Data Export

Export of data from most maintenance objects is supported via "Generalized" export. See Generalized Data Export in the *Application Framework Administrative User Guide* for more information about generalized data export.

Data Export Control Records

Data export from a maintenance object requires a Data Export Control record configured for that maintenance object. See **Enabling Data Export** in Generalized Data Export in the *Application Framework Administrative User Guide* for more information.

Change Data Capture for Measuring Components and Usage Subscriptions

For most maintenance objects, changes are tracked via the Capture Change for Ongoing Export audit algorithm (F1-MO-REGCHN).

Measuring Components and Usage Subscriptions use a "Most Recent" date field to track related transactions (Initial Measurements and Measurements in the case of Measuring Components and Usage Transactions in the case of Usage Subscriptions). These "Most Recent" fields are updated with each new initial measurement or usage transaction created by the system.

In order to prevent the Change Data Capture process from capturing changes based on these frequent transactions (daily for measurements and monthly for usage transactions), the Capture Change for Ongoing Export audit algorithm has been disabled on the Measuring Component and Usage Subscription maintenance objects.

Enabling change data capture and generalized data export for these maintenance objects requires adding the “Generalized Data Export BO Change Data Capture” (D1-BO-REGCHN) Audit algorithm to the appropriate Measuring Component and Usage Subscription business objects. This algorithm creates records in the Data Export Update table used by the generalized export process.

Measurement Data Export

Export of data from the Measurement maintenance object is supported via "Specialized" export. See Specialized Data Export in the *Application Framework Administrative User Guide* for more information about specialized data export.

The Measurement maintenance object is configured to support data export via the following options:

Option Type	Option Value
Data Export Class	F1SE (Specialized)
Export Batch Control	D1-SMDEC (Specialized Measurement Data Extract Current)

Current vs. Historical Extract

There are two types of extracts used with the Measurement maintenance object:

- **Daily Current Extracts** gather measurement data for measurements within the last 30 days that have been either added or changed. This extract is based on the **Measurement Date/Time** and **Last Update Timestamp** fields on the Measurement table, as well as the **Export Through Date/Time** field on the Data Export Control record (see below) to include only changes since the last extract.
- **Daily Historical Extracts** gather measurement data for measurements older than the last 30 days that have either been added or changed. To avoid having to scan the entire Measurement table, this extract is based on Change Data Capture. During initial measurement data processing if an initial measurement has measurements that are older than 30 days, then a Change Data Capture record is created in the General Process table.
  Enabling historical change data capture requires configuration of the following algorithm parameters:
  - Capture Historical Measurement Changes: true
  - Historical Measurement Data Extract Batch Control: D1-SMDEH (Specialized Measurement Data Extract Historical)

  These parameters should be configured as described on the following algorithms:
  - Update Latest Measurement Date/Time on MC with Consumption Sync (D1-UPD-DTMC)
  - Update Latest Measurement Date/Time on Scalar MC with Consumption Sync (D1-UDTSCMCRE)
  - Update Latest Date/Time on MC from a Future Measurement (D1-UPD-DTMCF)
  - Update Latest Date/Time on Scalar Measuring Component (D1-UPDDTSCMC)
  - Re-derive other values (D1-REDEROVAL)

Note: This use of Change Data Capture is an exception to the general rule described in the Specialized Data Export section of the *Application Framework Administrative User Guide*.

Data Export Control Records

Data export from the Measurement maintenance object requires a Data Export Control record. This record designates that the Measurement maintenance object should be exported, and tracks the date and time through which data has been exported so far, allowing subsequent export processes to only export incremental changes.

See Defining Data Export Control in the *Application Framework Administrative User Guide* for more information about creating Data Export Control records.

Batch Controls

Export of data and creation of export files is performed by the following batch controls:

- **Specialized Measurement Data Extract Current** (D1-SMDEC): Used for daily extracts of current (new) measurement data. This process extracts data that has changed since the last extract process has run (based on the **Export Through Date/Time** field on the Data Export Control record. This process updates the **Export Through Date/Time** field at the start of batch processing to eliminate the possibility of not setting an initial measurement as historical by Change Data Capture during the processing window.
- **Specialized Measurement Data Extract Historical** (D1-SMDEH): Used for daily extracts of historical measurement data. This process runs against Change Data Capture records in the General Process table.

Batch processing is threaded by on Measuring Component ID, and creates one or more files based on the number of threads processed and batch control parameters.

File Creation and Naming

The extract process uses two controls to determine the number of files to be created:

1. Each thread will create at least 1 file (more threads result in smaller files)
2. The **File Size Limit** parameter can be used to specify set the number of measurement rows to include in any given file. This can result in a single thread producing multiple files.

The data in each file is not ordered, each thread will retrieve measurement data for a set of Measuring Component IDs, but beyond that there will be no control for how the data is loaded into the file.

File Naming

Since each thread can produce multiple files and the number of files per thread cannot be predicted the file names will also include an indicator for the first file in a thread and the last file in a thread:

- MSRMT_EXPORT_<file prefix batch parameter>_<batch number>_<batch thread number>_<total thread count>_<thread file indicator>_<process date time>.csv.gz

where:

- <file prefix batch parameter> is an optional file prefix defined using the **File Name Prefix** batch control parameter.
- <batch number> is the batch number for the process
- <batch thread number> is the thread number that created the file
- <total thread count> is the total number of threads
- <thread file indicator>: is an identified for the first and last files created by a thread, S for first file E for last file
- <process date time>is the date and time at which the file was created, in YYYYMMDDhhmmss format (YYYY = year, MM = month, DD = day, hh = hour, mm = minute, ss = second)
- \

Examples: For batch number 135, comprising 10 threads,

- First file of 3rd out of 10 threads: MSRMT_EXPORT_PREFIX_135_3_10_S_20191104173233.csv.gz
- All files before the last file of 3rd out of 10 threads: MSRMT_EXPORT_PREFIX_135_3_10_20191104173233.csv.gz
- Last file of 3rd out of 10 threads: MSRMT_EXPORT_PREFIX_135_3_10_E_20191104173233.csv.gz

File Format

The extract process produces comma separated values (CSV) files with the data described in the following table. Note: all date/times will be converted from OUAF format to the appropriate ISO format in UTC with the time zone provided (as "Z").

				Applicability		
Field	**Data Type**	**Sample Content**	**Description**	**Interval**	**Scalar**	**Subtractive Interval**
MEASR_COMP_ID	CHAR(12)	000004769213	Primary Key of Channel from Meter (aka Device)	X	X	X
MSRMT_DTTM	DATE	2019-01-01T00:00:00Z	End of the time period the measurement covers. For example, if this were an hourly interval it would cover 2018-12-31T23:00:00Z to 2019-01-01T00:00:00Z	X	X	X

				Applicability		
Field	**Data Type**	**Sample Content**	**Description**	**Interval**	**Scalar**	**Subtractive Interval**
PREV_MSRMT_DTTM	DATE	2018-12-31T00:00:00Z	For scalar measurements this identifies the start of the period being measured. Note: there should be another measurement for the same MC with this date as the MSRMT_DTTM		X	
MSRMT_COND_FLG	CHAR(6)	501000	Defines the quality of the measurement (e.g. estimated, regular, super). Valid values defined on F1_EXTENDABLE_LOOKUP_VAL for the BUS_OBJ_CD D1-MeasurementConditionLookup	X	X	X
MSRMT_USE_FLG	CHAR(4)	D101	Indicates whether the measurement is usable Valid values defined on CI_LOOKUP		X	
USER_EDITED_FLG	CHAR(4)	D1UE	Indicates whether the user has edited the measurement. Valid values defined on CI_LOOKUP	X	X	X
READING_VAL	NUMBER(16,6)	750000000	This is the Dial Reading which is used to calculate the consumed amount.		X	X
READING_COND_FLG	CHAR(6)	501000	Defines the quality of the Dail Reading			X

				Applicability		
Field	**Data Type**	**Sample Content**	**Description**	**Interval**	**Scalar**	**Subtractive Interval**
			(e.g. estimated, regular, super). Valid values defined on F1_ EXTENDABLE_ LOOKUP_VAL for the BUS_ OBJ_CD D1-MeasurementConditionLookup			
COMBINED_ MULTIPLIER	NUMBER(18,6)	1	Contains the product of the Measuring Component Multiplier and the Installation Constant. The MSRMT_VAL will have been multiplied by this number.	X	X	X
MSRMT_VAL	NUMBER(16,6)	1.23	This is the consumption reported from the meter after going through VEE (aka billing ready)	X	X	X
MSRMT_VAL1	NUMBER(16,6)		This and the subsequent measurement values (MSRMT_ VAL 1 through 10) are derived values that are computed from the core MSMRT_VAL. Examples of this are UOM conversion (CCF to THERM) or loss adjustment.	X	X	X
MSRMT_VAL2	NUMBER(16,6)			X	X	X
MSRMT_VAL3	NUMBER(16,6)			X	X	X
MSRMT_VAL4	NUMBER(16,6)			X	X	X
MSRMT_VAL5	NUMBER(16,6)			X	X	X
MSRMT_VAL6	NUMBER(16,6)			X	X	X

Field	Data Type	Sample Content	Description	Applicability		
				Interval	Scalar	Subtractive Interval
MSRMT_VAL7	NUMBER(16,6)			X	X	X
MSRMT_VAL8	NUMBER(16,6)			X	X	X
MSRMT_VAL9	NUMBER(16,6)			X	X	X
MSRMT_VAL10	NUMBER(16,6)			X	X	X
CRE_DTTM	DATE	2019-10-31T11:09:31Z	When the record was created.	X	X	X
LAST_UPDATE_DTTM	DATE	2019-11-09T16:35:02Z	When the record was last updated.	X	X	X

Sample File

The following is a sample export file for interval data spanning a six hour window. Note that this does not include labels or column headers, but the data is in the same order as the column descriptions above in the **File Format** section above.

```
000004769213,2019-01-01T08:00:00Z,,501000,,,,1,1.3715,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:39:09Z,2019-08-09T20
000004769213,2019-01-01T09:00:00Z,,501000,,,,1,1.882,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:39:09Z,2019-08-09T20:
000004769213,2019-01-01T10:00:00Z,,501000,,,,1,1.838,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:39:09Z,2019-08-09T20:
000004769213,2019-01-01T11:00:00Z,,501000,,,,1,1.8585,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:39:09Z,2019-08-09T20
000004769213,2019-01-01T12:00:00Z,,501000,,,,1,1.836,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:39:09Z,2019-08-09T20:
000004769213,2019-01-01T13:00:00Z,,501000,,,,1,1.83,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:39:09Z,2019-08-09T20:3
000671510494,2019-01-01T08:00:00Z,,501000,,,,1,0.486,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:56:47Z,2019-08-09T20:
000671510494,2019-01-01T09:00:00Z,,501000,,,,1,0.4965,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:56:47Z,2019-08-09T20
000671510494,2019-01-01T10:00:00Z,,501000,,,,1,0.45,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:56:47Z,2019-08-09T20:5
000671510494,2019-01-01T11:00:00Z,,501000,,,,1,0.4725,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:56:47Z,2019-08-09T20
000671510494,2019-01-01T12:00:00Z,,501000,,,,1,0.4575,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:56:47Z,2019-08-09T20
000671510494,2019-01-01T13:00:00Z,,501000,,,,1,0.447,0,0,0,0,0,0,0,0,0,0,2019-08-09T20:56:47Z,2019-08-09T20:
020643301975,2019-01-01T08:00:00Z,,501000,,,,1,0.015,0,0,0,0,0,0,0,0,0,0,2019-08-09T21:04:28Z,2019-08-09T21:
020643301975,2019-01-01T09:00:00Z,,501000,,,,1,0.015,0,0,0,0,0,0,0,0,0,0,2019-08-09T21:04:28Z,2019-08-09T21:
020643301975,2019-01-01T10:00:00Z,,501000,,,,1,0.015,0,0,0,0,0,0,0,0,0,0,2019-08-09T21:04:28Z,2019-08-09T21:
020643301975,2019-01-01T11:00:00Z,,501000,,,,1,0.015,0,0,0,0,0,0,0,0,0,0,2019-08-09T21:04:28Z,2019-08-09T21:
020643301975,2019-01-01T12:00:00Z,,501000,,,,1,0.0165,0,0,0,0,0,0,0,0,0,0,2019-08-09T21:04:28Z,2019-08-09T21
020643301975,2019-01-01T13:00:00Z,,501000,,,,1,0.015,0,0,0,0,0,0,0,0,0,0,2019-08-09T21:04:28Z,2019-08-09T21:
```

Consumption Extract Type

Understanding Consumption Extract Type

The Consumption Extract Type controls which service point's measurements are extracted for Oracle Utilities DataConnect, what type of measurement is extracted, and how the measurements are grouped into TOU periods (if preferred).

The Consumption Extract Type also controls the request type used for creating Consumption Extract Requests of this type, how frequent automated requests are created for incremental extract, the batch jobs that are triggered for extracting data and the algorithms that extract, format, and write the data that is extracted.

Refer to the Oracle Utilities DataConnect integration section for more details on how this object is specifically put into use.

Configuring Consumption Extract Type

This portal is used to display and maintain a Consumption Extract Type.

Refer to Understanding Consumption Extract Types for more information.

You can access the portal by selecting **Admin**, then **Integration**, then **Consumption Extract Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Consumption Extract Type List**: displays all of the Consumption Extract Types so you can choose the one you want to display in more detail
- **Consumption Extract Type**: shows the specific configuration for the selected Consumption Extract Type

Analytics Configuration

This portal provides a bird's eye view of the configuration information for Oracle Utilities Analytics. It provides links and guidelines for the areas that need configuration to successfully run the extract transfer and load processes from Oracle Utilities Analytics.

Refer to the *Oracle Utilities Analytics Administration Guide* for more information.

You can access the portal from the **Admin > Analytics Configuration > Analytics Configuration**.

The following zones may appear as part of the portal's **Main** tab page

- **Bucket Configuration List.** This zone lists bucket configurations to set up.
- **BI-Related Business Objects Information.** This zone lists additional control entities to set up.

The following zones may appear as part of the portal's **OWB-Based ETL** tab page

- **Outbound Sync BOs and Algorithm List.** This zone lists business objects and algorithms involved in the extract process.
- **BI-Oriented Extendable Lookup.** This zone lists extendable lookups to set.
- **External Data Source Indicators List.** This zone lists the external identifiers of the various sources for your analytics data.

Note: Service Point business objects can make use of the following System Events:

- **Service Point Snapshot**: This system event defines the algorithm used to create a snapshot of the Service Point for use with Oracle Utilities Analytics. The Algorithm Entity for available algorithms is "SP (BO) - Snapshot."
- **Usage Snapshot**: This system event defines the algorithm used to create a usage snapshot of the Service Point for use with Oracle Utilities Analytics. The Algorithm Entity for available algorithms is "SP (BO) - Usage Snapshot."
- **Unreported Usage Analysis Snapshot**: This system event defines the algorithm used to create a snapshot of the Service Point's consumption since the last usage transaction for use with Oracle Utilities Analytics. The Algorithm Entity for available algorithms is "SP (BO) - Unreported Usage Analysis Snapshot."
- **SP VEE Exception Snapshot**: This system event defines the algorithm used to create a snapshot of VEE exceptions for the Service Point for use with Oracle Utilities Analytics. The Algorithm Entity for available algorithms is "SP (BO) - VEE Exception Snapshot."

Information Lifecycle Management (ILM)

Understanding Information Lifecycle Management (ILM)

Information Lifecycle Management (ILM) is a separately licensable component of Oracle Utilities Meter Data Management that works in tandem with Oracle database feature of the same name.

For further background on the overall ILM process see the Information Lifecycle Management chapter of the Oracle Utilities Application Framework *Administrative User Guide*.

This section provides an overview of the components used by the Information Lifecycle Management functionality, including:

- ILM-Enabled Maintenance Objects: Single Retention Period
- ILM-Enabled Maintenance Objects: Multiple Retention Periods
- Adjusting Eligibility of Non-Final Transactions
- Archive Eligibility Hierarchy
- Multiple Retention Period Eligibility Crawling Strategies

Refer to the Information Lifecycle Management chapter of the Oracle Utilities Application Framework *Administrative User Guide* for general information about how ILM works with OUAF applications.

ILM-Enabled Maintenance Objects: Single Retention Periods

The majority of Maintenance Objects support a single retention period. They can either inherit the system wide retention days that are configured on the ILM Configuration master configuration or they can have a retention period specified via the Maintenance Object option ILM Retention Period in Days.

For these objects either the Maintenance Object specific retention period or the system wide retention period will apply to all transactions.

ILM-Enabled Maintenance Objects: Multiple Retention Periods

For several Maintenance Objects the retention period may vary based on the type of transaction. For example, initial measurement data for non-billed units of measure such as voltage might be archived much more rapidly than measurements that feed into the production of billing determinants.

These maintenance objects are differentiated from the single retention period maintenance objects in a few ways:

- The primary table for the entity has an additional column Retention Period (RETENTION_PERIOD) that captures the retention period in days for each transaction. This value is populated based on the ILM Configuration - MDM master configuration.
- There is additional configuration available on the ILM Configuration -MDM master configuration to define multiple retention periods by maintenance object specific criteria
- Initial Measurement Data: retention periods can be specified by type of IMD (Interval vs Scalar), and primary UOM of the measuring component
- Device Events: retention periods can be specified by device event type category
- Activities: retention periods can be specified by activity type category
- There is a special ILM Crawler batch that supports crawling transactions by their specific retention periods
- A combination of each transactions ILM Date and Retention Period are considered when identifying records to evaluate for eligibility. Refer to batch controls ILM Crawler - Device Event (D1-DECRL), ILM Crawler - IMD (D1-IMDCL), or ILM Crawler - Activity (D1-ACTCR) for more details.
- There is an additional maintenance object option ILM Date Partition Months which feeds into the ILM Crawler batch to delay processing until all records on a partition would be eligible. This is explained later in this section under the header Multiple Retention Period Eligibility Crawling Strategies.

Adjusting Eligibility of Non-Final Transactions

As delivered Oracle Utilities Meter Data Management and Customer to Meter maintenance objects allow non-final transactions to be eligible for archive. This is a logical approach when transactional data has a very long retention period

(e.g. 3 year old non-final Initial Measurement Data is unlikely to have relevance and in many circumstances shouldn't be finalized).

If non-final transactions should not be eligible for archive this can be controlled at either the Maintenance Object (MO) or Business Object (BO) level:

- **MO - ILM Restrict By Status (Y/N)**: Setting this to "Y" opts into the ability to restrict archiving by status, this is required to be set for either of the MO or BO level restrictions to be considered. To enforce the status it requires that either the **ILM Restrict By BO Final Status (Y/N)** or **ILM Final Status Field Value** options be configured as well.
- **MO - ILM Restrict By BO Final Status (Y/N)**: Setting this to "Y" indicates the transactions for the MO must be in a final status to be eligible for ILM. A status is considered to be final based on the BO lifecycle configuration. This can be overridden by the BO option **Final Status Required for Archive (Y/N)**. This is only applicable for an MO that is maintained by business objects.
- **BO - Final Status Required for Archive (Y/N):** Setting this to "Y" indicates the transactions for this particular BO must be in a final status. This setting overrides the MO level **ILM Restrict By BO Final Status (Y/N).**

The following table helps to illustrate how these three settings impact whether a given transaction must be in a final status:

MO: ILM Restrict By Status	MO: ILM Restrict By BO Final Status	BO: Final Status Required for Archive	Final Status Required for Eligibility?
N	Y	Y	N
N	Y	N	N
N	N	Y	N
Y	N	N	N
Y	Y	N	N
Y	Y	Y	Y
Y	N	Y	Y
Y	Y	Not Provided	Y
Y	N	Not Provided	N

NOTE: There are a few MOs that do not support the BO level setting. To best understand how these fields are interpreted for an MO refer to the detailed description of the algorithm plugged into the ILM Eligibility system event for the MO. For example, the generic ILM Eligibility Based on Status (F1-ILMELIG) algorithm does not support the BO level override option.

Archive Eligibility Hierarchy

The table below summarizes details of the eligibility algorithms for each maintenance object supported by ILM. Information on this table includes:

- Cascaded By: Indicates how archiving for transactions for the maintenance object is initiated. For example, device events can be archived by themselves or as part of a related activity.
- ILM Date Computation: Indicates the type(s) of transactions that impact the calculation of the ILM Date for transactions for the maintenance object. "Itself" indicates that the ILM date is not impacted by other transactions.
- Specific Eligibility Considerations: Indicates specific considerations used by the eligibility algorithm.

- Cascaded Transactions: Indicates other transactions that are archived when transactions for the maintenance object are archived. For example, VEE exceptions are archived with initial measurements.

Maintenance Object	Cascaded By	ILM Date Computation	Specific Eligibility Considerations	Cascaded Transactions
Initial Measurement		Create date	VEE Exceptions must be eligible Associated Related MC Synchronization activity must be complete	VEE Exceptions
VEE Exception	Initial Measurement	Earlier of related Initial Measurement or VEE Exception Create Date	Related Service Issue Monitor must be complete	
Usage Transaction		The latest create date for all related usage transactions	All related usage transactions must be eligible for archiving	Related Usage Transactions
Device Event	Activity	Create date Superseded by paired event create date if earlier (where applicable)	Related paired events must be eligible Related service issue monitor must be complete Related activity must be complete	
Activity	Parent activity	Create date Superseded Device event, child activity, or completion event create date supersede (where applicable)	Cascaded transactions must be eligible Command requests must not be associated to a non-final service issue monitor Command requests must not be associated to an install event on off history entry	Child activities Device events Outbound communications Inbound communications Completion events
Outbound Communication	Activity	Create Date Superseded by initiating activity's create date (where applicable)		Inbound communication
Inbound Communication	Outbound Communication	Create date Superseded by initiating outbound communication create date (where applicable)		Device events

Proactive Archive Eligibility for Transactional Data

Some types of transaction data, including Device Events, Initial Measurement Data, and Usage Transactions can be analyzed for archive eligibility as they are received rather than waiting until their retention period has expired. This means that these records are judged ready to be archived as they are created and do not need to be read, analyzed, and updated during the ILM crawling process. This greatly reduces the number of records to be crawled at the end of the retention period

and shortens the window of time required for ILM crawling. This section outlines how the ILM Archive switch (ILM_ARCH_SW) is set to “Y” (Yes) for records of these types as part of their initial processing.

- **Device Events**:
  - The ILM Archive switch is set to “Y” for device events that meet the following criteria:
    - No service issue monitors were created for the device event
    - The device event is not a paired device event
  - The ILM Archive switch is set to “Y” for device events via the following algorithms:
    - Create Service Issue Monitor from Device Event (D1-DVCEVTSIM)
    - Create Service Issue Monitor from Device Event for Paired Event (D1-PRDVEVSIM)
- **Initial Measurement Data**:
  - The ILM Archive switch is set to “Y” by default for initial measurements (this is a different approach than the other record types).
  - The ILM Archive switch is set to “N” for initial measurements that meet the following criteria:
    - A consumption synchronization activity has been created for the initial measurement
    - A service issue monitor initiated from a VEE exception related to the initial measurement
  - The ILM Archive switch is set to “N” for initial measurements that meet the above criteria via the following algorithms:
    - Update Latest Measurement Date/Time on Scalar MC with Consumption Sync (D1-UDTSCMCRE)
    - Update Latest Measurement Date/Time on MC with Consumption Sync (D1-UPD-DTMC)
    - Create Service Issue Monitor from VEE Exception (D2-VEEEXCSIM)
- **Usage Transactions**:
  - The ILM Archive switch is set to “Y” when a usage transaction enters the “Sent” or “Discarded” state (or the “Calculated” or “Discarded” states for a sub-usage transaction) via the following algorithms:
    - Send Usage (D2–SEND-USG)
    - Set ILM Switch to ‘Y’ (D1-SETILMSWY)

Multiple Retention Period Eligibility Crawling Strategies

When multiple retention periods are defined for a maintenance object each record will have its retention period written to the database at the time of record creation. This retention period is then used to identify when the record will be ready for eligibility evaluation (aka crawling).

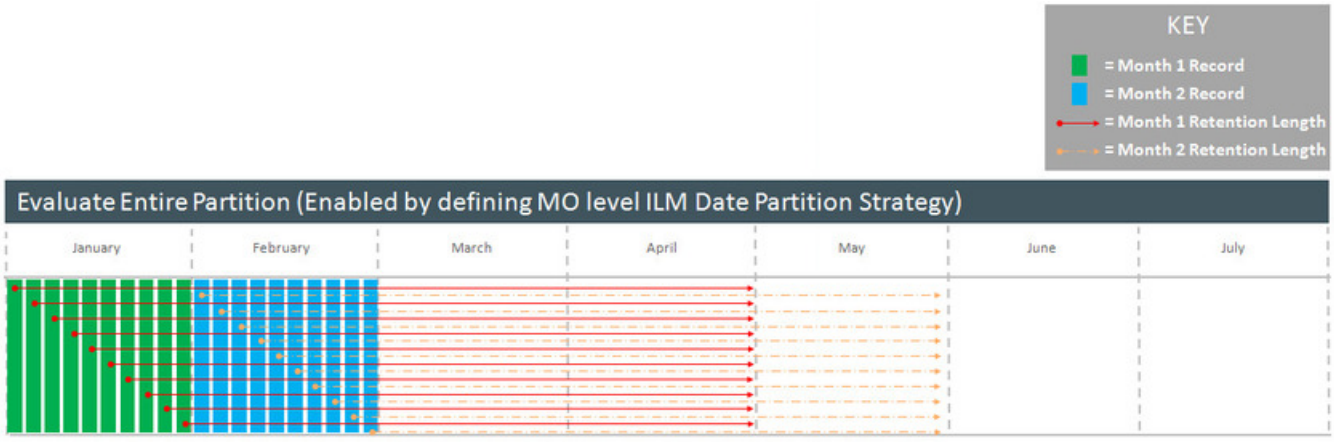
NOTE: As specified in the Oracle Utilities Meter Data Management Database Administrator's Guide each unique retention period for the maintenance object will be a sub-partition within a given ILM date partition.

The system supports two methodologies for doing so:

- Evaluate Individual Records:
- This is our default approach
- This is the standard approach to evaluating ILM retention for all maintenance objects with a single retention period
- The ILM date for each record will be compared against the current date less the retention period for that type of record. Simply stated, any records older than the retention period will be evaluated for eligibility.
- In this option we will process a portion of a partition each day and only once we had processed the entire partition would the DBA be able to take an archiving action on the partition

- This results in many visits to the partition to determine eligibility
- The image below illustrates how the evaluate individual records method will crawl one day of records on a partition and sub-partition each day until the retention period for the last day of the partition had expired. All records would then be archived together when the last day of the partition was marked as eligible. This results in daily processing of small amounts of records but frequent visits to the partition.

- Evaluate the Entire Partition
- This option will be enabled by configuring the ILM Date Partition Months in the maintenance object options
- In this option all records on a ILM date partition retention period sub-partition will be evaluated at the same time.
- For example, for a monthly partitioning strategy in the month of January we would not evaluate any records in the January partition until January 31st had reached the end of its retention period.
- This means that we would evaluate the 1st through the 31st all at one time
- This option should typically result in a single visit to the partition to determine eligibility
-
- The image below illustrates how the evaluate entire partition method will only crawl records from a given partition and sub-partition when all records on that partition are eligible to be evaluated. Therefore, when the last day of the partition is ready to be evaluated the entire partition will be evaluated. This results in periodic processing with, and in many cases, a single visit to the partition for ILM purposes.



Configuring Information Lifecycle Management (ILM)

For additional information on configuring ILM, see Enabling ILM for Supported Maintenance Objects in the Oracle Utilities Application Framework *Administrative User Guide*.

The tasks described in the OUAF guide must be applied in conjunction with the task of configuring the Oracle Utilities Meter Data Management specific master configuration.

You can access the portal from the **Admin > General > Master Configuration**.

Once the Master Configuration screen returns, locate the following record:

- **ILM Configuration - MDM:** this enables multiple retention periods to be configured for Initial Measurements, Device Events, and Activities.

Related Eligibility Algorithms

There are a several Oracle Utilities Meter Data Management specific algorithms involved with ILM refer to the detailed description of the algorithm type of each algorithm for more information about the functionality each provides:

1. **ILM Eligibility - Activity** (D1-ILMELGACT): determines eligibility of Activities and cascades child Activities, Communication Out, Communication In, Device Events, and Completion Events.
2. **ILM Eligibility - Communication In** (D1-ILMELGCI): determines eligibility of Communication In.
3. **ILM Eligibility - Communication Out** (D1-ILMELGCO): determines eligibility of Communication Out and cascades Communication In.
4. **ILM Eligibility - IMD** (D1-ILMELGIMD): determines eligibility of Initial Measurement data and cascades VEE Exceptions.
5. **ILM Eligibility - Device Event** (D1-ILMELGDE): determines eligibility of Device Event data and cascades VEE Exceptions.
6. **ILM Eligibility - Usage Transaction** (D1-ILMELIGUT): determines eligibility of Usage Transactions.
7. **ILM Eligibility - Usage Transaction Exception** (D1-ILMELGUEX): determines eligibility of Usage Transaction Exceptions.
8. **ILM Eligibility - VEE Exception** (D1-ILMELGVEX): determines eligibility of VEE Exceptions.

The following provides additional information beyond that provided in the ILM Eligibility - Activity (D1-ILMELGACT) algorithms' detailed descriptions on how specific activity types are handled:

- Request Orchestration Activities: Request orchestrations also archive any child activities that were created, and those child activities archive any child activities or data (completion events, device events, etc.).
- Field Activities / Command Activities: Field activities and command request activities are archived either as part of a request orchestration or by themselves. When a field activity/command request is archived, it also archives any of the following child transactions:
  - Update/Cancel Orchestrators
  - Communication Out
  - Communication In
  - Completion Event
- Non-Dispatchable Activities: Non-Dispatchable activities archived either as part of a request orchestration or by themselves. When a non-dispatchable activity is archived it also archives child completion events.

- Orchestration Maintenance Activities: Orchestration Maintenance activities are only archived by themselves if they do not have a related activity. When an orchestration maintenance activity is archived it also archives any of the following child transactions:
  - Communication Out
  - Communication In
- Device Event Activities: Presently device event activities are limited to outages which have an initiating and an ending device event. These types of activities cannot be archived until their related device events are either archived or ready to archive. However the device events do not have to wait for the activity to archive so the device events are not updated with the activity's ILM date.
- Bulk Activities: Bulk Activities are comprised of the Bulk Header and the Bulk Request/Response, and depending on how the header was created there will be one header to many bulk requests/responses. The Bulk Header will only be eligible for archive if all related Bulk Request activities are also eligible for archiving. All related bulk activities are archived together. Bulk activities also generate one-to-many command request activities. Those individual command activities are archived separately.
- Extract Request Activities: These activities are used to request data from the head-end system on a periodic basis. Extract request activities should be in a final state prior to being archived.
- Other Activities: Other types of activities can be archived provided they do not require any special logic for handling for archiving purposes, such as checking for related data. These activity types can archive without checking related transactions:
  - Consumption Sync
  - Dimension Scanner
  - Error Activity
  - Measurement Quantity (deprecated)
  - Meter Read Download Activity
  - Suppression
  - Usage Transaction Correction Processor

In the event that your implementation uses custom activity types that require special handling, a custom algorithm should be created and added prior to the base package algorithm to preemptively handle the activity type category.

Related Batch Controls

There are a several Oracle Utilities Meter Data Management specific batch controls involved with ILM:

1. **ILM Crawler - Activity** (D1-ACTCR): identifies and executes eligibility evaluation for Activities. This batch control supports multiple retention periods.
2. **ILM Crawler - Communication In** (D1-CICRL): identifies and executes eligibility evaluation for Communication In.
3. **ILM Crawler - Communication Out** (D1-COCRL): identifies and executes eligibility evaluation for Communication Out.
4. **ILM Crawler - Device Event** (D1-DECRL): identifies and executes eligibility evaluation for Device Events.
5. **ILM Crawler - IMD** (D1-IMDCL): identifies and executes eligibility evaluation for Initial Measurement Data. This batch control supports multiple retention periods.
6. **ILM Crawler - Usage Transaction Exception** (D1-UEXCL): :identifies and executes eligibility evaluation for Usage Transaction Exceptions.
7. **ILM Crawler - Usage Transaction** (D1-UTCRL): identifies and executes eligibility evaluation for Usage Transactions.
8. **ILM Crawler - VEE Exception** (D1-VEXCL): identifies and executes eligibility evaluation for VEE Exceptions.

For additional details see the Batch Processes section the Oracle Utilities Application Framework *Administrative User Guide*.

Measurement Reprocessing

Configuring Measurement Reprocessing

This section describes the process for configuring Measurement Reprocessing.

Refer to the About Measurement Reprocessing section of the *Oracle Utilities Meter Solution Business User Guide* for more functional information.

Activity Type Configuration

Configuration for Measurement Reprocessing is performed by adding or changing an Activity Type. You can access the portal from the **Admin > Communication > Activity Type**.

Once the Activity Type search screen returns, locate the following records:

- **Measurement Reprocess Activity - Interval:** this Activity Type handles measurement reprocessing for interval Measuring Components.
- **Measurement Reprocess Activity - Scalar:** this Activity Type handles measurement reprocessing for scalar Measuring Components.

Use the Add button beside the record to configure for the first time. If a record has already been added, then click the Edit button instead. Use the embedded help to guide you through the meaning of each configuration field. By adding these Activity Types you are activating the process within Oracle Utilities Meter Data Management that will monitor changes to either the Measuring Component multiplier changes or Install Event installation constant. If either of these attributes change for a device, then a new Activity will be created that attempts to reprocess the measurements for the affected period.

Related Batch Controls

There are a few batches involved with Measurement Reprocessing:

- **Measurement Reprocessing Monitor** (D1-MRAC): this processes any new Measurement Reprocessing activities that are created.
- **Measurement Reprocessing - Retry Monitor** (D1-MRER): this process retries any Measurement Reprocessing activities that fail to the Issue Detected state.

Outage Storm Mode

Configuring Outage Storm Mode

This section describes the process for configuring Outage Storm Mode.

NOTE: Refer to the About Outage Storm Mode section of the *Oracle Utilities Meter Solution Business User Guide* for more functional information.

Master Configuration settings

Configuration for Outage Storm Mode is performed by adding or changing a Master Configuration. You can access the portal from the **Admin > General > Master Configuration**.

Once the Master Configuration search screen returns, locate the **MDM Master Configuration**. Use the Add button beside the record to configure for the first time. If a record has already been added, then click the Edit button instead. The **Estimation Eligibility For Widespread Outages** section is the primary section that needs to be configured for Outage Storm Mode to be enabled. Use the embedded help to guide you through the meaning of each configuration field.

Setup Meter Communication Tracking Aggregation

As part of Outage Storm Mode, aggregations are performed to determine the percentage of meter communication that has occurred for an area. The following configuration supports this:

1. Access the portal for Measuring Component Type through **Admin > Device > Measuring Component Type**. Add a new Measuring Component Type using the **Meter Communication Tracking Aggregator Type** (D2-MtrCommTrckngAggregatorType) business object. Follow the embedded help provided for each section. Two areas listed below are crucial to preparing for this specific aggregation:
    a. When selecting a Measuring Component Business Object for this record, choose the **Meter Communication Tracking Aggregator** Business Object.
    b. For the Value Identifiers, there are a few metrics that are can be defined as listed on the D2-AGG-MCTM Algorithm Type:
        a. Read Percentage
        b. Count of Received Measuring Components
        c. Count of Total Measuring Components
2. Access the portal for Activity Type through Admin > Communication > Activity Type. Add a new record for the Meter Communication Tracking Dimension Scanner Activity Type.
3. Access the portal for Activity through **Main > Communication > Activity**. Add a new record for the Activity Type you defined in the prior step.

Standard Event Name Configuration

If there is a desire to indicate that a widespread outage has ended for a device when it receives certain device events, the **Standard Event Name** Extendable Lookup can be used for this. You can access the portal from the **Admin > General > Extendable Lookup**.

Any Device Event that should close out Estimation Suppression Activities must be configured with the "End Estimation Suppression" algorithm (D2-EN-ESTSUP) in the Additional Processing Algorithms section. This is often configured for "Power Up" events to indicate that normal power flow has resumed for the meter.

Related Batch Controls

There are a few batches directly involved with Outage Storm Mode:

- **Dimension Scanner** (D1-ADS): monitors the "Meter Communication Tracking Dimension Scanner" business object. New Aggregator Measuring Components will be created for every applicable Service Type, Postal Code, and Head End in the system.
- **Aggregation Monitor** (D2-AGG): runs to execute the logic for the "Meter Communication Tracking Aggregator" Measuring Component. This logic will aggregate all meters for the defined Service Type, Postal Code, and Head End. Once the read percentage is found for that day then a measurement will be created for the Aggregator Measuring Component.

- **Estimation Suppression Monitor** (D1-ESTSU): This batch monitors the Estimation Suppression activities for processing. This batch should be scheduled to run regularly so it can clean up Suppression Activities if they have ended.
- **Widespread Outage** (D2-WSO): This batch queries the Meter Communication Tracking Aggregators to determine if any have experienced a widespread outage. For any actively in outage mode, then the batch will find all related Service Points and execute the logic on the algorithm plugged into the BO system event of "Estimation Suppression". It's recommended that D2-WSO be scheduled shortly after aggregation D2-AGG.

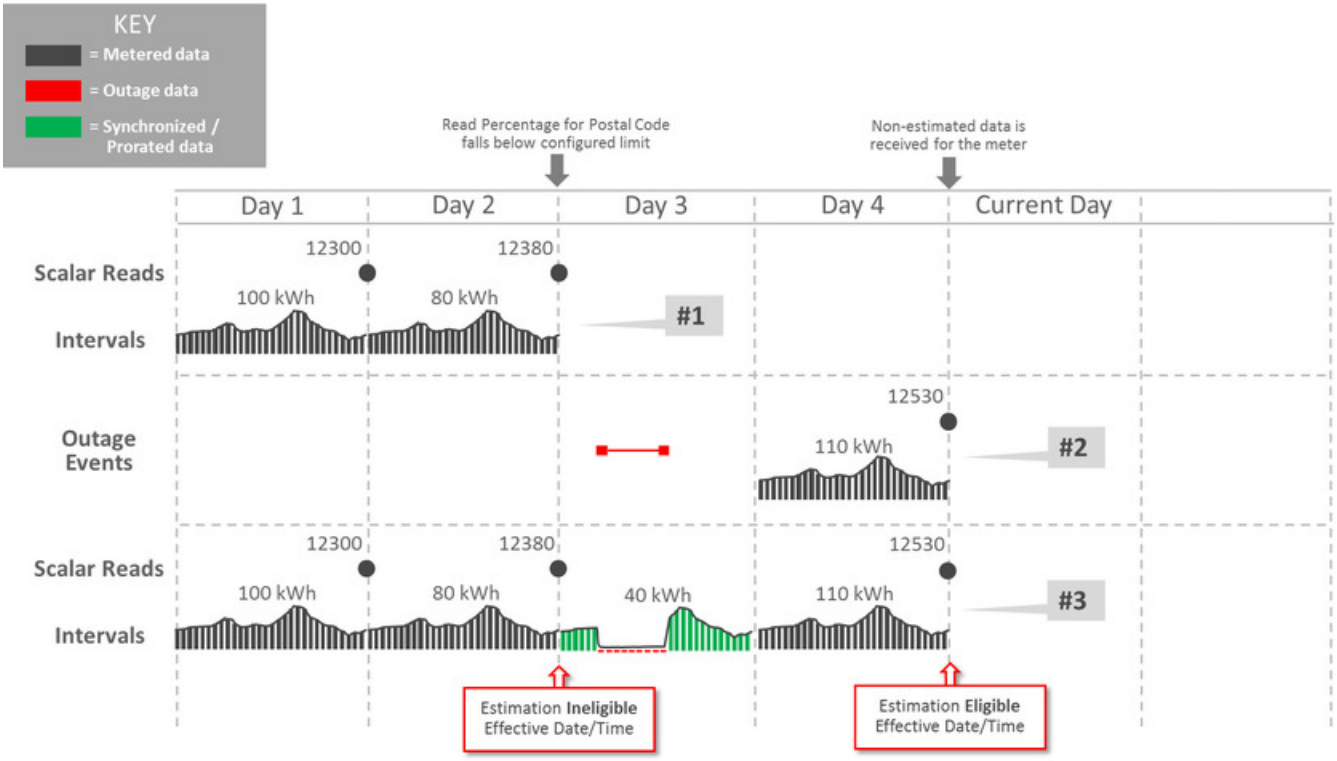
Detailed Examples of Outage Storm Mode

Scenario 1: Meter is marked estimation ineligible then received metered data with outage events

The following configuration settings should be assumed for this example:

Master Config option	Value
Check Widespread Outages	Yes
Read Percentage Days to Examine	7
Percent Drop From Norm - Estimation Ineligible	30.0%
Minimum Device Count For Ineligibility	100
Days Before Becoming Ineligible	0
Fill Missing Data With Zero	No
Days Before Filling Zero	
Maximum Estimation Ineligibility Days	3
Percent Within Norm - Resume Estimation	5.0%

Diagram:



Explanation for diagram:

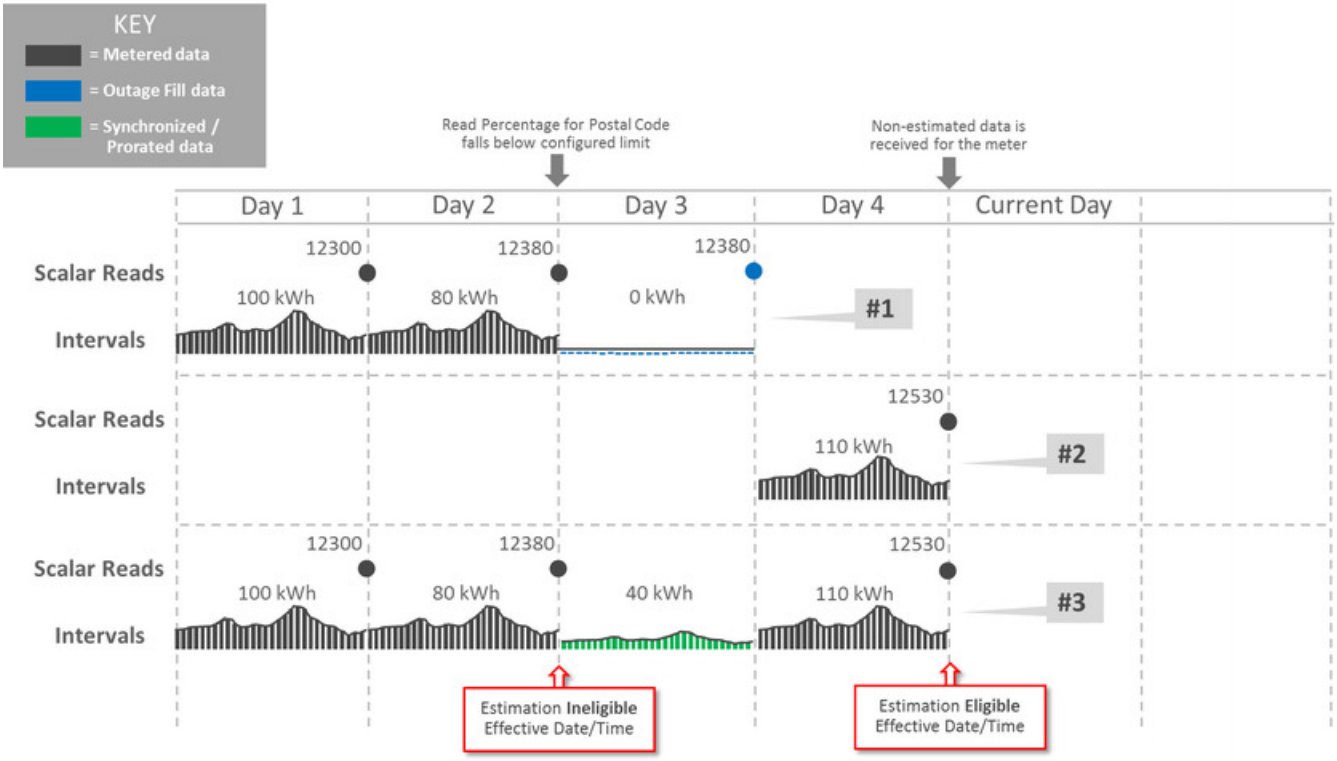
1. The meter was communicating consistently but then a widespread outage occurs in the same postal code so the meter has an estimation suppression created.
2. Two events are received...a Power Down event and a Power Up event. Also, Regular usage data is received from the meter. The reception of non-estimated data marks the meter as eligible for estimation again.
3. Oracle Utilities Meter Data Management estimates data and takes the two events into account. Intervals marked as outage and filled with zeros are created during the outage period by the D1-SMMTR batch.

Scenario 2: Meter is marked estimation ineligible, is filled with zeros, then receives actual data

The following configuration settings should be assumed for this example:

Master Config option	Value
Check Widespread Outages	Yes
Read Percentage Days to Examine	7
Percent Drop From Norm - Estimation Ineligible	30.0%
Minimum Device Count For Ineligibility	100
Days Before Becoming Ineligible	0
Fill Missing Data With Zero	**Yes**
Days Before Filling Zero	**0**
Maximum Estimation Ineligibility Days	3
Percent Within Norm - Resume Estimation	5.0%

Diagram:



Explanation for diagram:

1. The meter loses communication and periodic estimation fills in outage intervals.
2. The meter reestablishes communication and sends in the latest readings and interval data.
3. Oracle Utilities Meter Data Management creates new data that's synchronized to fall in line with the new data from the meter by the Consumption Sync batch. This data replaces the outage measurements as it balances out to the scalar and interval data.

Chapter 20

Integrations

External Applications

Understanding External Applications

External applications are applications and systems that are external to the Oracle Utilities meter data products, and can include customer information systems such as Oracle Utilities Customer Care and Billing, outage management systems such as Oracle Utilities Network Management System, or other types of applications.

External system service providers utilize processing methods to specify how the system sends and creates data used by the two applications. For example, when Oracle Utilities Meter Data Management is integrated with Oracle Utilities Customer Care and Billing, an external system representing Oracle Utilities Customer Care and Billing would specify how usage requests are received and processed by Oracle Utilities Meter Data Management.

Refer to Understanding Process Methods for more information about processing methods.

External Applications Impact Data Import and Export

External Applications are configured to identify how a particular external system communicates data with Oracle Utilities Meter Data Management. This includes:

- The identifier type used to locate devices and measuring components. These are used both on import and export of data.
- The date/time format used in various data imports (i.e. whether or not the date/time format includes time zone information).

Please refer to the embedded help for more information about these fields.

Each external application can be associated to an external system which is used to define the messages that can be sent to that service provider and how each message is sent.

Configuring External Applications

This portal is used to display and maintain External Applications.

Refer to Understanding External Applications for more information.

You can access the portal from the **Admin**, then **Integration**, then **External Application**.

The following zones may appear as part of the portal's **Main** tab page:

- **External Applications List:** This zone lists all External Application records. Broadcast a record to display the details of the selected record.
- **External Application:** This zone provides information about the selected External Application.
- **Processing Method List:** This zone provides the list of processing methods defined for the External Application.
- **Translation Method List:** This zone provides the list of translation methods defined for the External Application.
- **Inbound BOs Send By Service Provider:** This zone lists inbound Business Objects that are sent by this External Application. The identification is driven by the Business Object having a Business Object Option of type "Sent By Service Provider" that references the current External Application.

Business Flags

Understanding Business Flags

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, Oracle DataRaker and Oracle Utilities Customer Care and Billing. 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 (Oracle DataRaker) may identify potential theft situations that should be investigated prior to sending any usage transactions.

The subsequent sections describe business flag functionality that is specific to the service point business flag as implemented by this product.

When creating a new device type there are the following options:

Name	Details	Business Object
Service Point Business Flag Type	Provides configuration for business flags associated to a service point. This type offers an ability to hold a business flag for manual analysis.	D1-SPBusinessFlagType
Service Point Monitor Business Flag Type	Provides configuration for business flags that should initiate service issue monitors and convey results of a field activity to the intended subscribers.	D1-SPMonitorBusinessFlagType

Refer to About Business Flags in the *Oracle Utilities Framework Administration User Guide* for detailed information about the components of business flags.

Important Business Flag Business Object Options

Below are a list of business object options that are defined on the business flag business object associated to a given business flag type:

- **Valid Action**: identifies a BPA script that can be used to guide the user though an action that can be taken on a business flag. Note: this option should be paired with a business object lifecycle option of "Execution of Valid Actions Permitted" with a value of "true" for any lifecycle status that should support the execution of valid actions.

More detail about these options can be found by visiting a measuring component business object and inspecting the business object options.

Field Activities 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 determine the confidence 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.

Refer to algorithm type **Assign Business Flag Final Confidence** (D1-ABFFC) for additional details.

Configuring Business Flags

For information on configuring business flags, see Setting Up Business Flag Configuration in the Oracle Utilities Application Framework *Administrative User Guide*.

Oracle Utilities Customer Care and Billing

Overview

This section provides an overview of how Oracle Utilities Meter Data Management supports integrations with a customer information system. In an integration between Oracle Utilities Meter Data Management (MDM) and a customer information system such as Oracle Utilities Customer Care and Billing (CC&B):

- Oracle Utilities Meter Data Management is typically the "system of record" for meter-related data, including meter records, meter configurations, validation, editing, and estimation (VEE) rules, bill determinant calculation rules, usage data, and calculated bill determinants.
- Oracle Utilities Customer Care and Billing (the customer information system) is typically the "system of record" for account related and service point-related data, including the rates and tariffs used to calculate bills for each account and customer.

Given this breakdown of data between the two systems, any integration between them must account for the passage of data between the two to ensure that each system can accurately perform its business functions.

Configuring Master Data Synchronization

In most integrations with Oracle Utilities Customer Care and Billing (or other CIS), Oracle Utilities Meter Data Management is not used as the system of record for account, customer, or service point-related data. Synchronizing this data between the two systems ensures that all account, customer, and service point-related data in Oracle Utilities Meter Data Management is correct and up to date before usage transaction calculations are performed. This synchronization process is supported through a set of business objects, master configurations, batch controls, and pre-configured Inbound Web Services.

- **Initial Synchronization Requests:** Initial synchronization requests are used when initially setting up Oracle Utilities Meter Data Management. They facilitate import of data that creates devices, device configurations, measuring components, service points, install events, contacts, and usage subscriptions in Oracle Utilities Meter Data Management based on corresponding data in Oracle Utilities Customer Care and Billing.
- **Ongoing Synchronization Requests:** Ongoing synchronization requests are used when updating existing data in Oracle Utilities Meter Data Management based on changes in corresponding data in Oracle Utilities Customer Care and Billing. Ongoing synchronization requests can be used to update contacts, devices, device configurations, measuring components, install events, service points, and usage subscriptions.
- **Composite Synchronization Requests:** Composite synchronization requests are requests that contain synchronization requests for multiple types of data within a single request. For example, a composite request could contain requests to update device, device configuration, measuring component, and install event data. This supports situations where multiple types of data must be updated based on a single change in Oracle Utilities Customer Care and Billing.

Master Configuration

The following master configurations are used in Oracle Utilities Meter Data Management to configure the sync process between CIS and MDM:

Master Configuration	Name / Description
Master Data Synchronization Configuration	Lists all foreign key references that need resolution. Each one should reference the view that contains the external key / production key cross-reference. For entities that undergo both the initial and the ongoing sync, two views are specified. For entities that undergo the ongoing sync, an external system / ID type mapping is specified to cater for entities that might be synchronizing from more than one external system.
Seeder Sync Request Master Configuration	Lists the maintenance objects (device, device configuration, etc.) that require synchronization. Each references the synchronization business object that needs to be instantiated when processing a synchronization request for that maintenance object. For maintenance objects that undergo both initial and the ongoing synchronization, two business objects are specified.

Inbound Data Synchronization Business Objects

The integration between Oracle Utilities Meter Data Management and Oracle Utilities Customer Care and Billing uses the following inbound (to Oracle Utilities Mater Data Management) synchronization business objects:

Object Name	Business Object	Description
Device Config Composite Sync Request	D1-CompositeSyncRequestDC	A composite synchronization request that handles syncing in Device Configuration

		and creating related Measuring Component. This business object a child of the D1-CompositeSyncRequest business object.
Contact Initial Sync Request	D1-InitialSyncRequestContact	Instances of this business object represent individual initial contact synchronization requests.
Device Configuration Initial Sync Request	D1-InitialSyncRequestDC	Instances of this business object represent individual initial device configuration synchronization requests.
Device Initial Sync Request	D1-InitialSyncRequestDevice	Instances of this business object represent individual initial device synchronization requests.
Install Event Initial Sync Request	D1-InitialSyncRequestIE	Instances of this business object represent individual initial install event synchronization requests.
Measuring Component Initial Sync Request	D1-InitialSyncRequestMC	Instances of this business object represent individual initial measuring components synchronization requests.
Service Point Initial Sync Request	D1-InitialSyncRequestSP	Instances of this business object represent individual initial service point synchronization requests.
Dynamic Option Initial Sync Request	D2-InitialSyncRequestDynOpt	Instances of this business object represent individual initial dynamic option synchronization requests.
Dynamic Option Event Initial Sync Request	D2-InitialSyncRequestDynOptEvt	Instances of this business object represent individual initial dynamic option event synchronization requests.
Usage Subscription Initial Sync Request	D2-InitialSyncRequestUS	Instances of this business object represent individual initial usage subscription synchronization requests.
Ongoing Sync Request Acknowledgement	D1-OngoingSyncReqAckMsg	Used to send a message to the sending system to acknowledge receipt of an ongoing synchronization request.
Scalar Meter Read Ongoing Sync Request	D1-OngoingSyncReqScalarMtrRead	Instances of this business object represent individual ongoing scalar meter read synchronization requests.
Contact Ongoing Sync Request	D1-OngoingSyncRequestContact	Instances of this business object represent individual ongoing contact synchronization requests.
Device Configuration Ongoing Sync Request	D1-OngoingSyncRequestDC	Instances of this business object represent individual ongoing device configuration synchronization requests.
Device Ongoing Sync Request	D1-OngoingSyncRequestDevice	Instances of this business object represent individual ongoing device synchronization requests.
Install Event Ongoing Sync Request	D1-OngoingSyncRequestIE	Instances of this business object represent individual ongoing install event synchronization requests.

Measuring Component Ongoing Sync Request	D1-OngoingSyncRequestMC	Instances of this business object represent individual ongoing measuring component synchronization requests.
Service Point Ongoing Sync Request	D1-OngoingSyncRequestSP	Instances of this business object represent individual ongoing service point synchronization requests.
Dynamic Option Ongoing Sync Request	D2- OngoingSyncRequestDynOpt	Instances of this business object represent individual ongoing dynamic option synchronization requests.
Dynamic Option Event Ongoing Sync Request	D2- OngoingSyncRequestDynOptEvt	Instances of this business object represent individual ongoing dynamic option event synchronization requests.
Usage Subscription Ongoing Sync Request	D2-OngoingSyncRequestUS	Instances of this business object represent individual ongoing usage subscription synchronization requests.
Sync Request Seeder	D1-SyncRequestSeeder	Used to identify the appropriate synchronization business object to create when processing synchronization requests.
Contact Synchronization Add	D1-SynchronizationAddContact	Used when adding a new contact as a result of a synchronization request.
Device Configuration Synchronization Add	D1-SynchronizationAddDC	Used when adding a new device configuration as a result of a synchronization request.
Device Synchronization Add	D1-SynchronizationAddDevice	Used when adding a new device as a result of a synchronization request.
Install Event Synchronization Add	D1-SynchronizationAddIE	Used when adding a new install event as a result of a synchronization request.
Measuring Component Synchronization Add	D1-SynchronizationAddMC	Used when adding a new measuring component as a result of a synchronization request.
Service Point Synchronization Add	D1-SynchronizationAddSP	Used when adding a new service point as a result of a synchronization request.
Usage Subscription Synchronization Add	D2-SynchronizationAddUS	Used when adding a new usage subscription as a result of a synchronization request.

Batch Controls

Batch controls perform processing for initial synchronization requests such as allocating keys to data, resolving foreign keys, and loading data (instantiating business objects representing entities such devices, measuring components, etc.).

"Initial Sync Request - Resolve Keys XXX" batch controls invoke a generic maintenance object transition process to invoke the "Resolve Keys - Initial Sync" algorithm for synchronization requests of the appropriate type. Parameters used by "resolve keys" batch controls include:

- **Maintenance Object**: (Required) the maintenance object (device, device configuration, etc.) to be processed. This must be set to the Sync Request maintenance object for the batch control (device for device synchronization requests, service point for service point synchronization requests, etc.)
- **Restrict By Batch Code**: Restricts processing to synchronization requests whose current state is linked to this batch code.
- **Restrict By Business Object**: Restricts processing to synchronization requests linked to this business object.

- **Restrict By Status Code**: Restricts processing to synchronization requests of this status (default: KEY_ALLOCATD).
- **Max Errors**: Specifies the maximum number of errors allowed before the process exits.

"Initial Sync Request - Load Data XXX" batch controls load data (created new instances of business objects) for requests of the appropriate type (device, measuring component, etc,). Parameters used by "load data" batch controls include:

- **Maintenance Object**: (Required) the maintenance object (device, device configuration, etc.) to be processed. This must be set to the Sync Request maintenance object for the batch control (device for device synchronization requests, service point for service point synchronization requests, etc.)
- **Restrict By Batch Code**: Restricts processing to synchronization requests whose current state is linked to this batch code.
- **Restrict By Business Object**: Restricts processing to synchronization requests linked to this business object.
- **Max Errors**: Specifies the maximum number of errors allowed before the process exits.

The table below lists the batch controls used by initial synchronization requests:

Batch Code	Name / Description
D1-CMSYN	Composite Sync Request
D1-SIIER	Initial Sync Request - Error
D1-SILCN	Initial Sync Request - Load Data Contact
D1-SILDC	Initial Sync Request - Load Data DC
D1-SILDV	Initial Sync Request - Load Data Device
D1-SILIE	Initial Sync Request - Load Data IE
D1-SILMC	Initial Sync Request - Load Data MC
D1-SILSP	Initial Sync Request - Load Data SP
D1-SILUS	Initial Sync Request - Load Data US
D1-SIKCN	Initial Sync Request - Resolve Keys Contact
D1-SIKDC	Initial Sync Request - Resolve Keys DC
D1-SIKDV	Initial Sync Request - Resolve Keys Device
D1-SIKIE	Initial Sync Request - Resolve Keys IE
D1-SIKMC	Initial Sync Request - Resolve Keys MC
D1-SIKSP	Initial Sync Request - Resolve Keys SP
D1-SIKUS	Initial Sync Request - Resolve Keys US
D1-SIOER	Ongoing Sync Request -Error
D1-SIOPE	Ongoing Sync Request - Pending
D1-SRSDE	Sync Request Seeder - Error
D2-SIKDO	Initial Sync Request - Resolve Keys DO
D2-SIKDE	Initial Sync Request - Resolve Keys DOE
D2-SIKUS	Initial Sync Request - Resolve Keys US
D2-SILDO	Initial Sync Request - Load Data DO
D2-SILDE	Initial Sync Request - Load Data DOE
D2-SILUS	Initial Sync Request - Load Data US
F1-SAKRQ	Sync Request Allocate Keys Monitor
F1-SRLRQ	Sync Request Load Records Monitor

F1-SYNRQ	Sync Request Monitor Process
F1-SYSRQ	Sync Request Sampling Monitor (Deferred)

Batch Control Scheduling

The following table specifies the order in which the batch controls on the Initial Sync Request BO life cycle should be executed. The first row identifies the maintenance object for which the synchronization request is intended and the first column specifies the type of process.

	Contact	Usage Subscription	Service Point	Install Event	Device Configuration	Device	Measuring Component	Dynamic Option	Dynamic Option Event
Transformation / Schema Validation Job	1	1	1	1	1	1	1	1	1
Key Allocation Job	2	2	2	2	2	2	2	2	2
Foreign Key Resolution / BO Validation Job (dependent on ALL Key Allocation Jobs finishing)	3	8	3	6	3	3	6	3	5
Load Job	4	9	5	7	5	4	7	4	6

Note that before the Key Resolution job is run, all the Key Allocation Jobs need to finish. This ensures that all foreign key references can be subsequently resolved.

Some business object-level validation is dependent on other entities being completely loaded first. The sequence numbers above allow for this. For example, usage subscriptions business object validation is dependent on service points existing; Install Event business object validation is dependent on both service points and devices existing.

Inbound Web Services

Inbound web services are used to facilitate invoking the Sync Request Seeder business object by the middleware components upon receipt of a synchronization request.

The table below lists the pre-configured Inbound Web Services used to process synchronization requests sent from Oracle Utilities Customer Care and Billing.

Inbound Web Service	Description	Schema Name
D1-SyncRequestInbound	Sync Request Inbound	D1-SyncRequestSeeder (BO)

Example Sync Process Steps

This section provides an overview of the processing that takes place when a synchronization request is sent. For each step in the process, the table below provides a brief description of the processing that takes place, and lists the specific objects involved.

Step	Process	Objects
1	Oracle Utilities Customer Care and Billing sends a synchronization request to the middleware integration layer. For example, consider a request to update information about a service point.	
2	The middleware components transforms the request from the Customer Care and Billing format, to the format used by Oracle Utilities Meter Data Management (this format is based on the business object schemas of the synchronization request business objects).	
3	The middleware component invoke the appropriate Inbound Web Service, and sends the transformed request.	Inbound Web Service: D1- SyncRequestInbound (mapped to the D1-SynrRequestSeeder business object)
4	The Inbound Web Service invokes the Sync Request Seeder business object, which in turn, determines which synchronization request business object to create (based on the type of data in the synchronization request and the Seeder Sync Master Configuration).	Synchronization Request BO: D1- OngoingSyncRequestSP
5	For initial synchronization requests, background processing creates master data for each synchronization request, including the following steps: • Data Transformation / Schema Validation • Allocate Keys • Resolve Foreign Keys / Validate Business Object • Load Data	
6	For ongoing synchronization requests, the following steps are performed by Enter algorithms on the synchronization request business object lifecycle states: • Data Transformed/Basic Schema Validated: • Determine Sync Request BO • Setup Transformed Data	Setup Transformed Data Algorithm: D1- SETTRANDT Sync Request Pre-Add Data Algorithm: D1-SR-PREADD Resolve Keys - Ongoing Sync Algorithm: D1-RESKEYFAL Sync Request Update Data Algorithm: D1- SR-UPDDAT

	• Pre-Added: • Sync Request Pre-Add Data • FKs Resolved: • Resolve Keys - Ongoing Sync • Updating: • Sync Request Update Data

Configuring the Bill Determinant Interface

Oracle Utilities Customer Care and Billing uses bill determinant data (usage transactions based on meter readings) calculated and stored in Oracle Utilities Meter Data Management when calculating bills for customers. This process allows Oracle Utilities Customer Care and Billing to send requests for usage transaction calculations to Oracle Utilities Meter Data Management, which in turn performs the requested calculations, and publishes the results back to Oracle Utilities Customer Care and Billing. Processing usage transaction requests is supported through a set business objects, pre-configured Inbound web services, and processing methods.

Business Objects

The table below lists the business objects used when processing usage transaction requests.

Business Object	Description
D2-UsgTranSeeder	Used to determine the usage transaction business object to use when creating new usage transactions.

Inbound Web Services

Inbound web services are used to facilitate invoking the Usage Transaction Seeder business object by the middleware components upon receipt of a usage transaction request, and to update the usage transaction upon completion of the bill.

The table below lists the pre-configured Inbound Web Services used to process usage transaction requests sent from Oracle Utilities Customer Care and Billing.

Inbound Web Service	Description	Schema Name
D2-UsageTransactionRequestInbound	Usage Transaction Request Inbound	D2-UsgTranSeeder (BO)
D2-UsageTransactionUpdateInbound	Update Usage Transaction Bill Flags	D2-UpdUTMul (Service Script)

Service Provider Processing Methods

Processing methods are used to determine the usage transaction business object to use when creating usage transactions based on the requests, and for determining the method by which usage transactions are sent back to Oracle Utilities Customer Care and Billing.

Processing Method	Business Object	Description
How To Send US Information - Online	D2-HowToSendUSInfoOnline	Used to specify the method (batch control, business object, or outbound message) by which usage transactions are sent to subscribing systems in real time.

How To Send US Related Information	D2-HowToSendUSInformation	How To Send US Related Information Used to specify the method (batch control, business object, or outbound message) by which usage transactions are sent to subscribing systems.
How To Create US Related Information	D2-HowToCreateUSInformation	Used to determine the type of usage transaction business object to create.
How To Send US Related Information Batch	D2-HowToSendUSInfoBatch	Used to specify the method (batch control, business object, or outbound message) by which usage transactions are sent to subscribing systems via batch processing.

Example Bill Determinant Process Steps

This section provides an overview of the processing that takes place when a usage transaction request is sent. For each step in the process, the table below provides a brief description of the processing that takes place, and lists the specific objects involved.

Step	Process	Objects
1	Oracle Utilities Customer Care and Billing sends a usage transaction request to the middleware integration layer.	
2	The middleware components transforms the request from the Customer Care and Billing format, to the format used by Oracle Utilities Meter Data Management (this format is based on the business object schemas of the synchronization request business objects).	
3	The middleware component invoke the appropriate Inbound Web Service, and sends the transformed request.	Inbound Web Service: D2- UsageTransactionRequestInbound (mapped to the D2-UsgTranSeeder business object)
4	The Inbound Web Service invokes the Usage Transaction Seeder business object. This business object: • Determines the usage subscription ID based on an external usage subscription ID • Determines the appropriate usage transaction business object to create	Processing Method: D2- HowToCreateUSInformation (How To Create Usage Subscription Related Information)
5	The usage transaction determines the usage calculation group(s) to use when calculating usage. These are retrieved from the usage subscription (with a fallback to the usage subscription type). This base package logic can be overridden by specifying an algorithm in the **Usage Calculation Group Determination Override Algorithm** field on the usage subscription type. If an algorithm is specified in this field,	

	its logic overrides the existing method of determining the usage calculation group. A base package algorithm (D2-DRVUSGGRP) is provided that can be used here. This algorithm uses the **CCB Rate Schedule** extendable lookup used to map the rate schedules in Oracle Utilities Customer Care and Billing to usage calculation groups. The algorithm's logic looks for a combination of rate history list entries on the usage subscription and the type of device configuration installed during the bill period to determine the usage calculation group(s) to use for bill determinants calculation.	
6	The usage transaction business object calculates usage based on the date range provided in the request.	
7	If the usage transaction has a sub usage transactions, it checks the status of each.	
8	If the usage transaction is configured to require an approval, a To Do Entry is created.	
9	The usage transaction then sends the usage transaction to any subscribing systems.	Processing Method: D2- HowToSendUSInformation (How To Send US Related Information)
10	Upon completion of the bill, Oracle Utilities Customer Care and Billing invokes an inbound web service to update the "Used on Bill" and/or "Linked to Frozen Bill Segment" fields as appropriate to indicate if the usage transaction was used on a bill.	Inbound Web Service: D2-UsageTransactionUpdateInbound (Update Usage Transaction Bill Flags)

Oracle Utilities Operational Device Management

Overview

This section provides an overview of how Oracle Utilities Meter Data Management supports integrations with Oracle Utilities Operational Device Management. In an integration between Oracle Utilities Meter Data Management and Oracle Utilities Operational Device Management:

- Oracle Utilities Meter Data Management is typically considered the "system of record" for service points (asset locations), contacts, and device installation information (install events)
- Oracle Utilities Operational Device Management is typically considered the "system of record" for assets (devices) and asset specification configuration (device configurations)

Given this breakdown of data between the two systems, any integration between them must account for the passage of data between the two to ensure that each system can accurately perform its business functions. The integration between Oracle

Utilities Meter Data Management and Oracle Utilities Operational Device Management is based on data synchronization between the two systems.

Configuring Master Data Synchronization

The specific data synchronization flows supported between Oracle Utilities Meter Data Management (MDM) and Oracle Utilities Operational Device Management (ODM) include the following:

- **Asset-Device Synchronization**: As new assets and asset specification configurations are created or changed in ODM, corresponding devices and device configurations must be created or changed in MDM
- **Service Point/Contact - Asset Location/Contact**: As Service Points and/or Contacts are created or changed in MDM, corresponding Asset Locations and Contacts must be created or changed in ODM
- **Install Events- Asset Location/Disposition**: As devices are installed/removed in MDM, corresponding changes to an asset's Disposition (location and status) must be made in ODM

This synchronization process is supported through a set of business objects, master configurations, batch controls, and pre-configured Inbound Web Services. Refer to the *Oracle Utilities Integration for Device Operations Implementation Guide* for more information about this integration and the data synchronization processes used by this integration.

Master Configuration

The following master configurations are used to configure the sync process between MDM and ODM:

Master Configuration	Name / Description
Master Data Synchronization Configuration	Lists all foreign key references that need resolution. Each one should reference the view that contains the external key / production key cross-reference. For entities that undergo both the initial and the ongoing sync, two views are specified. For entities that undergo the ongoing sync, an external system / ID type mapping is specified to cater for entities that might be synchronizing from more than one external system.
Seeder Sync Request Master Configuration	Lists the maintenance objects (device, device configuration, etc.) that require synchronization. Each references the synchronization business object that needs to be instantiated when processing a synchronization request for that maintenance object. For maintenance objects that undergo both initial and the ongoing synchronization, two business objects are specified.
ODM Integration Master Configuration	Specifies the external system used to represent Oracle Utilities Operational Management, and the URL for the Oracle Utilities Operational Management application. Also specifies the outbound message types used to send synchronization requests to Oracle Utilities Operational Management.

Inbound Data Synchronization Business Objects

The integration between Oracle Utilities Meter Data Management and Oracle Utilities Operational Device Management uses the following inbound (to Oracle Utilities Mater Data Management) synchronization business objects:

Object Name	Business Object	Description
Device Ongoing Sync Request	D1-OngoingSyncRequestDevice	Used to synchronize devices and device configurations in Oracle Utilities Mater

		Data Management based on assets and asset specification configurations created/ updates in Oracle Utilities Operational Device Management.

Outbound Data Synchronization Business Objects

The integration between Oracle Utilities Meter Data Management and Oracle Utilities Operational Device Management uses the following outbound (from Oracle Utilities Mater Data Management) synchronization business objects (based on the F1-SYNC REQ maintenance object)

Object Name	Business Object	Description
ODM Sync Request	D1-ODMContactSyncRequest	Used to synchronize contacts in Oracle Utilities Operational Device Management based on contacts created/updated in Oracle Utilities Meter Data Management
ODM Install Event Sync Request	D1-ODMInstallEventSyncRequest	Used to synchronize asset location/disposition in Oracle Utilities Operational Device Management based on install events created/ updated in Oracle Utilities Meter Data Management
ODM SP Sync Request	D1-ODMSPSyncRequest	Used to synchronize asset locations in Oracle Utilities Operational Device Management based on service points created/updated in Oracle Utilities Meter Data Management

Oracle Utilities Analytics

Overview

Oracle Utilities Meter Data Management can also be integrated with Oracle Utilities Analytics (OUA) to allow users to view analytic data based on usage, events, and other data tracked in Oracle Utilities Meter Data Management.

Oracle Utilities Meter Data Management Analytics comprises the following products:

- **Oracle Utilities Meter Data Analytics**: dashboards, dashboard pages, and analytics used to view usage, event, and other data from Oracle Utilities Meter Data Management
- **Oracle Utilities Meter Data Management Extractors and Schema**: the star schema and extract programs used by Oracle Utilities Meter Data Analytics.

Configuring Extracts

Oracle Utilities Meter Data Management includes the following components used when integrating with Oracle Utilities Analytics:

Master Configuration

The following master configurations are used to configure the sync process between MDM and OUA:

Master Configuration	Description
Analytics Master Configuration - ODI-Based	This master configuration is primarily used to configure the information that will be used in the extraction of data for business intelligence when using the Oracle Data Integrator extract, transform, and load (ETL) process.

Batch Controls

Batch Code	Name / Description
D1-SPSDL	Service Point Snapshot Download
D2-DUGDL	Usage Subscription Usage Calculation Group Download
D2-MVREF	Materialized View Refresh
D2-USGDL	Service Point Usage Snapshot Download
D2-UUSDL	SP Unreported Usage Snapshot Download
D2-VEEDL	SP VEE Exception Snapshot Download

Refer to the following documentation for more information about Oracle Utilities Analytics:

Oracle Utilities Extractors and Schema for Oracle Utilities Meter Data Management Data Mapping Guide

Oracle Utilities Analytics Dashboards for Meter Data Analytics Metric Reference Guide

Oracle Utilities Analytics Admin Guide

Oracle DataRaker

This section describes integrations with Oracle DataRaker.

Business Flag Integration

With the business flag integration with Oracle DataRaker:

- Either Oracle Utilities Meter Data Management or Oracle DataRaker can initiate a business flag
- Oracle Utilities Meter Data Management can notify Oracle DataRaker of a change in status of a business flag:
  - if a business flag in error has been discarded
  - if a business flag's confidence has been updated
  - with the results of field work initiated by a business flag

For additional details see About Business Flags in the Oracle Utilities Application Framework *Administrative User Guide*.

Configuring Business Flag Integration

Oracle Utilities Meter Data Management includes the following components used when integrating with Oracle DataRaker:

Master Configuration

The following master configurations are used to configure the sync process between MDM and Oracle DataRaker:

Master Configuration	Description
DataRaker Integration Master Configuration	This master configuration is used to enable deep linking into Oracle DataRaker. Multiple URL patterns are supported with an ability to provide substitution variables to satisfy the URL parameters (e.g. "&searchTech="). This also provides a mapping between Oracle Utilities Meter Data Management service types and the Oracle DataRaker point types.

Service Provider Processing Methods

The following processing methods should be associated to the service provider configured for Oracle DataRaker:

Processing Role	Description
Business Flag Confidence Change	This role will identify the appropriate message to send when a business flag confidence has been changed (e.g. from Suspected to Confirmed). When not configured no message will be sent.
Business Flag Discarded Notification	This role will identify the appropriate message to send when a business flag has been discarded. When not configured no message will be sent.
Business Flag Field Work Details	This role will identify the appropriate message to send when a business flag has completed that initiated field work. When not configured no message will be sent.
Business Flag Added Notification	This role will identify the appropriate message to send when a business flag has been created from within Oracle Utilities Meter Data Management. When not configured no message will be sent.

Extendable Lookups

The following extendable lookups should be reviewed for Oracle DataRaker:

Extendable Lookup	Description
Business Flag Standard Name	If Oracle DataRaker will be communicating business flags using a non-standard name provide an entry in the Valid External Name with the non-standard name for each standard name communicated by Oracle DataRaker.

Usage and Event Integration

Oracle Utilities Smart Grid Gateway adapters can be configured to publish initial measurement and device event data for use in Oracle DataRaker or other external systems. Published data is posted to a user-defined directory that Oracle DataRaker can monitor for import.

Initial measurement and device event data is published in the “native” data format (the format of the initial measurement and device event seeder business objects). This format includes normalized unit of measure, condition, and device event

codes. See the *Oracle Utilities Smart Grid Gateway Adapter Development Kit Administrative User Guide* for more details about this format.

NOTE: Initial measurement data published via this feature is published prior to VEE processing. In addition, filtering can NOT be applied to initial measurement or device event data published via this feature.

Configuring DataRaker Usage and Event Integration

Oracle Utilities Smart Grid Gateway adapters use OSB and BPEL components when publishing initial measurement and device event data for integration with Oracle DataRaker.

OSB Projects

The following "CM" Oracle Service Bus projects are used to configure the integration between SGG and Oracle DataRaker:

Adapter	OSB Projects
Landis+Gyr	SGG-D3-USAGE-CM (used to publish usage)
	SGG-D3-EVENT-CM (used to publish device events)
NES	SGG-D4-USAGE-CM (used to publish usage)
	SGG-D4-EVENT-CM (used to publish device events)
MV90	SGG-D5-USAGE-CM (used to publish usage)
Sensus	SGG-D6-USAGE-CM (used to publish usage)
	SGG-D6-EVENT-CM (used to publish device events)
Silver Spring Networks	SGG-D7-SSNXML-CM (used to publish usage)
	SGG-D7-CSV-CM (used to publish device events)
Itron OpenWay	SGG-D8-ITRONXML-CM (used to publish usage)
	SGG-D8-EXCEPTION-CM (used to publish device events)
Adapter Development Kit	SGG-DG-SEEDER-CM (used to publish usage and device events)

The following components provided with these projects are used in publishing measurement data and device events to Oracle DataRaker:

- The **DataRakerBusinessService** business service is used to send data to a pre-configured JMS queue (defined as an Endpoint URI), from which the data will be published. This business service is specified in the EnvironmentSettings.xq file (see below).
- The **DataRakerServiceAccount** service account is used to define and maintain the user name and password needed to access the JMS queue defined in the **DataRakerBusinessService** business service.

Enabling Data Publishing

Publishing initial measurement and device event data is enabled by referencing a publisher business service (DataRakerBusinessService) in the publishServices/service element in the EnvironmentSettings.xq files provided with the OSB projects listed above as follows:

```
<publishServices>
  <service>[publisherBusinessService]</service>
</publishServices>
```

BPEL Composites

The SGGDRIntegration BPEL composite handles publishing the data to Oracle DataRaker or other systems.

Configuring Publishing Output

The following parameters can be used to configure details of how the data is provided to Oracle DataRaker, including the directory where files are posted for Oracle DataRaker to consume, number of records per file, polling frequency, etc. These parameters are defined during installation. See the *Oracle Utilities Smart Grid Gateway Installation Guide* for more details about defining values for these parameters.

Parameter	Description	Default Value
SGG_DR_INT_QUEUE	JNDI name of queue to publish SGG payloads. This is the JMS queue defined in the **DataRakerBusinessService** business service. This should NOT be changed.	DataRakerQueue
SOA_DR_PUBLISH_SIZE	The number of records (SGG payloads) to accumulate in a published file.	100
SOA_DR_FILE_SIZE	The maximum file size for the accumulated (SGG payloads) file in kilobytes.	524288
SOA_DR_ELAPSED_TIME	The period of time in second which, when exceeded, causes a new outgoing file to be created.	600
SOA_DR_POLLING_FREQ	The polling frequency in seconds of the staging directory for new files.	60
SOA_DR_STAGING_DIR	Mount point/directory for the staging directory for accumulated SGG payload files. This is used internally and should NOT be changed.	/spl/sploutput/staging
SOA_DR_INTEGRATION_DIR	Mount point/directory from which Oracle DataRaker will consume the converted XML files.	/spl/sploutput/int

Oracle Utilities Network Management System

Overview

This section provides an overview of how Oracle Utilities Meter Data Management supports integrations with Oracle Utilities Network Management System. In an integration between Oracle Utilities Meter Data Management and Oracle Utilities Network Management System:

- Oracle Utilities Smart Grid Gateway can notify Oracle Utilities Network Management Systems of device events that have been received (e.g. last gasp and power up).
- Oracle Utilities Network Management Systems can request:
- Oracle Utilities Smart Grid Gateway to perform various smart meter commands for a device or list of devices (e.g. device status check)
- Oracle Utilities Smart Grid Gateway to suppress device event notifications for a certain set of device event types.

Configuring Device Event Notification Suppression

Oracle Utilities Meter Data Management includes the following components used when integrating with Oracle Utilities Network Management Systems:

Service Provider Processing Methods

The following processing methods should be associated to the service provider configured for Oracle Utilities Network Management System:

Processing Role	Description
Bulk Response	This role will identify the appropriate message to send with the results of a bulk command execution. For example, this would be used if a large list of devices had been supplied to execute device status check in order to verify outage status.
Response - Success	This role will identify the appropriate message to send for a successful smart meter command
Send Device Event	This role sets up the subscription for device events that Oracle Utilities Network Management System has interest in.
Suppression Notification	This role identifies the appropriate message to send when a notification has been set or removed.

Activity Types

The following processing activity types should be configured:

Activity Type	Description
Bulk Request Header	Configuration for the bulk request header activity type which is used in making smart meter command requests for a list of meters.
Bulk Response	Configuration for the bulk response activity type which organizes the response for bulk requests.
Device Event Notification Suppression	Identifies the device event categories and types that are subject to suppression.

Oracle Utilities Customer Self Service

Overview

Oracle Utilities Meter Data Management can be integrated with Oracle Utilities Customer Self Service (OUCSS) to allow utility customers to:

- View their usage data
- Create self-service meter readings (i.e. submit their own meter readings)

- Perform rate comparisons (via integration between CC&B and MDM)

Master Configuration

The following master configurations are used to configure the sync process between MDM and OUCSS:

Master Configuration	Description
Self Service Master Configuration	This master configuration is used for general configuration related to the MDM to OUCSS interface. This includes a variety of configuration including: temperature source, rate comparison, supported usage calculation groups, etc.

Supporting Oracle Utilities Meter Data Management Services

Oracle Utilities Meter Data Management provides the following services used by Oracle Utilities Customer Self Service:

Service Name	Description	Logic	Inbound Web Service
Create Self-Service Meter Read	Allows users to submit their own meter reads via the Customer Self Service application.	Service Script: WX-CrSSMRead	WX-CreateSelfServiceMeterRead
Get Scalar Consumption Summary	Retrieves consumption data for display in the Customer Self Service application.	Service Script: WX-GetSCsum	WX-GetScalarConsumptionSummary
Get Usage Overview	Retrieves an overview of a customer's usage for a user-specified duration for display in the Customer Self Service application	Service Script: WX-GetUsgOVw	WX-GetUsageOverview
Multiple Account TOU Usages by Service Request Type	Uses the Get Usage Details service to retrieve usage for a list of accounts for display in the Customer Self Service application. Each account's usage is summarized by service type, UOM and SQI.	Service Script: WX-MulAccTOU	WX-MultipleAccountTOUUsagesByServiceType
Multiple Account Usages by Service Request Type	Uses the Get Usage Details service to retrieve usage for a list of accounts for display in the Customer Self Service application. Each account's usage is summarized by service type, UOM and SQI.	Service Script: WX-MulAccCmp	WX-MultipleAccountUsagesByServiceType
Multiple Account Usage Download Request	Uses the Get Usage Details service to retrieve usage for a list of accounts (by usage subscription) for display in the Customer Self Service application.	Service Script: WX-MulAccUsg	WX-MultipleAccountUsagesDownload
Get Usage Details	Retrieves usage details for a customer for a user-specific time period (year, month, day)	Business Service: WX-RETWSSTOUMapping	WX-RETWSSTOUMappingService

	for display in the Customer Self Service application		
Usage Adjustment Retrieval	Retrieves usage adjustment types based on a user-specified rate schedule. Used by the Rate Compare feature of the Customer Self Service application.	Service Script: WX-RetUsgAdj	WX-RetUsgAdj OR WX-UsageAdjustmentRetrieval

These services are based on the service scripts and business services noted above, and are invoked via the corresponding Inbound Web Services.

Refer to the *Oracle Utilities Customer Self Service Implementation Guide* for more information about integrating Oracle Utilities Meter Data Management with Oracle Utilities Customer Self Service.

Configuring Rate Compare Usage Adjustment Profiles

This section describes the steps involved in setting up usage adjustment profiles in Oracle Utilities Meter Data Management for use with the Rate Compare feature in Oracle Utilities Customer Self- Service.

This section provides an overview of the steps involved in setting up usage adjustment profiles. The sections that follow provide additional details and examples for each type of data that must be configured.

1. Create a characteristic type and values for each type of usage adjustment to be made available for rate comparison purposes. For example, if you wanted to allow adjustments for installation of solar panels, use of an energy-efficient appliance, or use of an electric vehicle, you would create three characteristic types (one for each type of adjustment), and values for each option within each type (each type of appliance or electric vehicle supported).
2. Create profile (stand-alone) measuring components that correspond to each characteristic type value.
3. Create profile data for each measuring component. This is interval data that represents the impact of the usage adjustment on a customer's consumption.
4. Create a profile factor for each usage adjustment type.
5. Create factor values for each factor. These values correspond to the characteristic type value, and link characteristic type values to profile measuring components.
6. Create one (or more) Profile Accumulation usage calculation rules that will apply the usage adjustment profile to the customer's interval consumption when calculating usage for the rate comparison request.
7. Create entries in the CCB Rate Schedule extendable lookup to associate the usage calculation group that contains the "Profile Accumulation" usage calculation rule to a rate schedule in Oracle Utilities Customer Care and Billing.
8. Create entries in the Usage Adjustment Types extendable lookup for each usage adjustment type.
9. Set up the Rate Compare Configuration section of the Self-Service Master Configuration to link usage adjustment factors and usage adjustment types to rate schedules Oracle Utilities Customer Care and Billing (defined in the CCB Rate Schedule extendable lookup).

Characteristic Types

Create a characteristic type and values for each type of usage adjustment you wish to make available to customer self-service users. For example, to create a usage adjustment profile for use of an electric vehicle, you would create an "electric vehicle" characteristic type and define values for each type of electric vehicle users can select.

Example Characteristic Type:

- Characteristic Type: ELEC_VEH
- Description: Electric Vehicles

- Type of Char Value: Predefined Value
- Characteristic Values:

Characteristic Value	Description
LEAF	Nissan Leaf
TESLA	Tesla Model S
VOLT	Chevrolet Volt

Profile Measuring Components

Create profile measuring components for each characteristic type value. These measuring components will be used to store profile data for each type of usage adjustment.

Note: The base package does not include stand-alone measuring component/measuring component type business objects, but the demonstration database contains "Standalone Interval" and "Standalone Interval Measuring Component Type" business objects that can be used to create profile measuring components and types.

Example Profile Measuring Component Type:

- Measuring Component Type: KWH-PROFILE
- Description: KWH Profile
- Measuring Component Business Object: Standalone Interval (demo)
- Measurement Business Object: Measurement
- Service Type: Electric
- Allow Negative Consumption: Allowed
- Consumptive / Subtractive: Consumptive
- Seconds Per Interval: 01:00:00
- Value Identifiers:
- Value Identifier Type: Measurement
- Short-Hand Description: kWh
- UOM: Kilowatt-Hours

Example "Nissan Leaf" Profile Measuring Component:

- Measuring Component Type: KWH Profile
- Number of Digits Left: 5
- Number of Digits Right: 5
- Time Zone: US Pacific Time
- Status: Active
- How To Use: Additive
- External ID: Nissan Leaf

Example "Tesla Model S" Profile Measuring Component:

- Measuring Component Type: KWH Profile
- Number of Digits Left: 5

- Number of Digits Right: 5
- Time Zone: US Pacific Time
- Status: Active
- How To Use: Additive
- External ID: Tesla Model S

Example "Chevrolet Volt" Profile Measuring Component:

- Measuring Component Type: KWH Profile
- Number of Digits Left: 5
- Number of Digits Right: 5
- Time Zone: US Pacific Time
- Status: Active
- How To Use: Additive
- External ID: Chevrolet Volt

Profile Data

Create profile data for each profile measuring component. This data represents the impact of the usage adjustment on a customer's consumption.

Note that this profile data can (and often will) include negative interval values to represent the difference in consumption applicable for the usage adjustment type. For example, if an energy- efficient electric clothes dryer uses an average of 30 kilowatt hours less per month than an average electric clothes dryer, profile data for that appliance might be a "straight line" hourly profile (a profile in which all intervals are of the same value) in which each value equals "0.042" (30 kWh per month divided by an average of 720 hours per month).

Factors / Factor Values

Create a profile factor for each usage adjustment type. This factor should use the "Factor Characteristic Source N/A Algorithm" to derive the appropriate characteristic type value based on the factor value, and should reference the characteristic type created earlier.

Example Electric Vehicle Factor:

- **Factor**: ELECTRIC_VEHICLE
- **Description**: Electric Vehicle
- **Factor Class**: Profile
- **Characteristic Source Algorithm**: Factor Characteristic Source N/A Algorithm
- **Factor Characteristic Type**: Electric Vehicle

Example "Nissan Leaf" Factor Value

- **Factor: Electric Vehicle**
- **Factor Characteristic Type: Electric Vehicle**
- **Factor Characteristic Value: LEAF**
- **Effective Date/Time: 01-01-2014 12:00:00AM**
- **Profile: Nissan Leaf, KWH Profile**

Example "Tesla Model S" Factor Value

- **Factor: Electric Vehicle**
- **Factor Characteristic Type: Electric Vehicle**
- **Factor Characteristic Value: LEAF**
- **Effective Date/Time: 01-01-2014 12:00:00AM**
- **Profile: Tesla Model S, KWH - 60 Minutes**

Example "Chevrolet Volt" Factor Value

- **Factor: Electric Vehicle**
- **Factor Characteristic Type: Electric Vehicle**
- **Factor Characteristic Value: LEAF**
- **Effective Date/Time: 01-01-2014 12:00:00AM**
- **Profile: Chevrolet Volt / KWH - 60 Minutes**

Profile Accumulation Usage Calculation Group and Rule

Create a usage calculation group that contains a "Profile Accumulation" usage calculation rule. This rule will calculate usage by accumulating historical usage with profile data. based on a selected profile factor value.

Note: Profile Accumulation rules should use eligibility criteria to ensure they are only executed when the "Calculation Mode" on the usage transaction is set to "Hypothetical Calculation" (D2HC).

Example: Profile Accumulation Usage Calculation Rule

- **Usage Calculation Group: Electric Residential Interval KWH**
- **Usage Calculation Rule: KWH_PROFILE_ACCUMULATION**
- **Sequence: 10**
- **Description: KWH Profile Accumulation**
- **Category: Usage Calculation**
- **Vector Source Configuration:**
- **Vector Type**: Channels Linked to Usage Subscription
- **Unit of Measure**: Kilowatt-Hours
- **Time of Use**:
- **Service Quantity Identifier**:
- **Target SPI**: 01:00:00
- **Result Processing Configuration**:
- **Apply TOU Map to Derived Vector: Yes**
- **TOU Map: Summer / Winter, 15 minute interval**
- **Result Storage Configuration**:
- **Insert Primary SQ Entry:** Yes
- **Save Derived Vector:** No
- **Service Quantity Identifier:**
- **Extract Interval Data:** No

CC&B Rate Schedule Extendable Lookup

Create an entry in the "CCB Rate Schedule" extendable lookup to associate the usage calculation group that contains the "Profile Accumulation" usage calculation rule to an applicable rate schedule in Oracle Utilities Customer Care and Billing.

Example CCB Rate Schedule extendable lookup:

- **Rate**: E-INT-RES
- **Description**: Electric Residential Interval Rate
- **Default Usage Calculation Group**: Electric Residential Interval KWH (E-INT-RES)

Usage Adjustment Types Extendable Lookup

Create an entry in the "Usage Adjustment Type" extendable lookup for each type of usage adjustment that will be available to customer self-service users. These entries are used in the Rate Compare Configuration section of the Self-Service Master Configuration (see below).

Example Electric Vehicle entry:

- **Usage Adjustment Type: ELEC_VEHICLE**
- **Description: Purchase of Electric Vehicle**
- **Override Description: Purchase of Electric Vehicle**
- **External Reference ID: Purchase of Electric Vehicle**

Self-Service Master Configuration - Rate Compare Configuration

Configure the "Rate Compare Configuration" section of the Self-Service Master Configuration to associate usage adjustment factors and usage adjustment types with an applicable rate schedule in Oracle Utilities Customer Care and Billing (defined in the CCB Rate Schedule extendable lookup).

Example Self-Service Master Configuration:

- **Factor Characteristic Type Indicating No Value Variation: N/A**
- **External Reference Factor Value Characteristic Type: External Reference ID**
- **Minimum Days of Usage Adjustment Data: 2**
- **Rate / Usage Adjustments:**
- **Rate**: Electric Residential - Interval KWH
- **Usage Adjustments**:
- Usage Adjustment Factor: Electric Vehicle
- Usage Adjustment Type: Purchase of Electric Vehicle

Oracle Utilities Dataconnect / Opower

Overview

Oracle Utilities DataConnect facilitates extraction of data from Oracle Utilities Meter Data Management 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.

Oracle Utilities DataConnect can be used to extract two types of data:

- Measurement data for service points based on the Service Point Types defined for a Consumption Extract Type. When extracting usage measurement data, Oracle Utilities DataConnect extracts both interval measurement data as well as frequently read scalar usage data (which is converted into interval measurements during the extraction process).
- Individual service points and install events (based on Service Point Types defined in an extract algorithm).

The extraction processes used for each type of data are separate, but extracted data can be correlated in external systems through service point data 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 Meter Data Management
- CIS Service Point ID: The ID used for the service in a customer information system (CIS) such as Oracle Utilities Customer Care and Billing

External systems receiving data extracted from Oracle Utilities DataConnect can use either (or both) of these IDs to associate extract measurement data to extracted master data.

Consumption Extract

When sending interval measurement usage 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 should include data history for all active service points for a specified historical period. Current data should be extracted on a regular ongoing (or incremental) basis. However, in addition to sending current data, any historical corrections received by the system should be extracted as well.

Extract Requests

There are several types of consumption extract requests:

- **Initial Load**: Initial load extract requests are created and submitted manually via the Consumption Extract Request portal. Consumption extract requests are based on a specified Consumption Extract Type (see below) and extraction date range. An initial load request must be created and submitted for each consumption extract type defined in the system.
- **Incremental / Ongoing (Current Data)**: Incremental / ongoing extract requests can be manually created, but more often will be created via a batch process. The "Create Daily Consumption Extract Requests" batch control scans active consumption extract types and creates a request for each one that has Frequency of "Automated Daily." Ad-hoc incremental requests can be created and submitted manually if needed.
- **Historical Correction**: Historical correction extracts are created via batch process. Algorithms on the Finalized state of the initial measurement and measurement business objects determine if a finalized initial measurement or rederived values are historical corrections. These algorithms create records which are evaluated by a batch process which extracts the measurements for the related initial measurements.

Consumption Extract Type

Consumption Extract Types define the specific parameters used when processing a consumption extract request. Consumption Extract Types control the service point type, type of measurement, and how the measurements are grouped into TOU periods if applicable. Consumption extract types also define the algorithm and batch processes to use when extracting data for different types of requests (initial load, incremental, and historical).

There are two consumption extract type business objects provided with the base package:

- **Energy Management Extract** (D2-ConsumptionExtractType): This business object retrieves interval data and converts it to a specified Target UOM and Interval Size. This business object does not support TOU mapping.

- **Energy Management Extract with TOU Mapping** (D2-ConsumptionExtractTypeTOU): This business object retrieves interval data, converts it to a specified Target UOM and Interval Size, and maps it to a specific TOU map prior to extraction.

Historical Versus Current Data

The "Extract Through Date/Time" field on the Consumption Extract Type is used to differentiate between current data (the most recently extracted data) and historical corrections, and is set to the last date on which data was extracted for that extract type. For example, if data is extracted on June 1, 2015, the "Extract Through Date/Time" would be set to "June 1, 2015 12:00AM." If/when data is extracted the next day, "Extract Through Date/Time" would be updated to "June 2, 2015 12:00AM."

When evaluating data for extract:

- Interval data is considered current if its measurement date time is after the "Extract Through Date/Time".
- Interval data is considered a historical correction if its measurement date time is on or before the "Extract Through Date/Time".

Historical data changes to an initial measurement can be detected when it enters the Finalized state. If the initial measurement is determined to be for a historical period by comparing its end date/time against the "Extracted Through Date/Time" on the Consumption Extract Type, a general process record will be written for the initial measurement so that measurements for it can be extracted. In addition, re-derived values on final measurements can also trigger the creation of a general process record for related initial measurements.

The following algorithms are used in this process:

- The "Create General Process Record if IMD is Historical Correction" algorithm is used to determine if a finalized initial measurement data is a historical correction. If it is, the algorithm creates a general process record for the initial measurement. This algorithm is provided in the base package, but not specified on initial measurement business objects by default. This algorithm should be defined as an Enter algorithm on the Finalized state of the initial measurement business objects.
- The "Create General Process Record for Re-derived Values" algorithm creates general process records for initial measurements associated with re-derived values. Processing will proceed as if a historical correction came in through an initial measurement. This algorithm is provided in the base package, but not specified on measurement business object by default. This algorithm should be defined as an Enter algorithm on the Re-derive state of the final measurement business object.

Consumption Extract Requests

Initial load and ongoing consumption extracts are created via Consumption Extract Requests. While extracts of these types can be created via adhoc submission of a batch job, requests are the preferred method for these types of consumption extracts.

The consumption extract request business object lifecycle includes logic that maintains and updates the "Extraction Through Date/Time" field on Consumption Extract Types, which is used to determine if daily requests should be created by the "Create Daily Consumption Extract Requests" batch control, and detect historical corrections.

A single business object is provided in the base package for consumption extract requests:

- **Consumption Request** (D2-IntervalDataExtRepository): This business object contains the information and lifecycle responsible for submitting the extract job, monitoring the run until it's finished, and then updating the Consumption Extract Type's "Extract Through Date/Time" on the Consumption Extract Type. This business object is based on the Request (F1-REQ) maintenance object.

The consumption extract process uses a set of base package algorithms to extract and format the interval data for export. These algorithms are specified in the "'Algorithms" section on the Consumption Extract Type as appropriate. These algorithms can be configured to allow for extraction of data for frequently-read scalar measuring components as well as interval measuring components. Frequently-read scalar measuring components are defined as scalar measuring components whose Read Method is set to "Automatic Read." When extracting measurements for frequently-read scalar measuring

components, scalar measurements are converted to interval measurements as part of the extraction process. This conversion uses the profile associated with the measuring component type. If no profile can be found, the interval data uses a flat line profile.

Initial Load / Incremental / Ongoing Requests: the following algorithms are used to extract and format interval data for initial load and incremental / ongoing requests:

- **Extract Initial Load/Ongoing Consumption for DataConnect** (D2-IDEXTPRD): This algorithm retrieves a service point's consumption for a given period and writes the results to a flat file.
- **Extract Tab-Delimited Initial Load/Ongoing Consumption (with UTC Offset)** (D2-IDEXTPRDU): This algorithm retrieves a service point's consumption for a given period and writes the results to a tab-delimited flat file. This algorithm also supports adjusting the consumption based on Daylight Saving Time.
- **Extract Initial/Ongoing Consumption and Apply TOU Map for DataConnect** (D2-IDEXTPTOU): This algorithm retrieves a service point's consumption for a given period, applies a TOU Map to the consumption, and writes the results to a flat file.
- **Extract Tab-Delimited Initial/Ongoing Consumption and Apply TOU Map (with UTC Offset)** (D2-IDEXTPTUU): This algorithm retrieves a service point's consumption for a given period, applies a TOU Map to the consumption, and writes the results to a tab-delimited flat file. This algorithm also supports adjusting the consumption based on Daylight Saving Time.

Historical Corrections: the following algorithms are used to extract and format interval data for historical correction requests:

- **Extract Historical Correction Consumption for DataConnect** (D2-IDEXTIMD): This algorithm retrieves historical correction consumption for a service point and writes the results to a flat file.
- **Extract Tab-Delimited Historical Correction Consumption (with UTC Offset)** (D2-IDEXTIMDU): This algorithm retrieves historical correction consumption for a service point and writes the results to a tab-delimited flat file. This algorithm also supports adjusting the consumption based on Daylight Saving Time.
- **Extract Historical Corrections and Apply TOU Map for DataConnect** (D2-IDEXTITOU): This algorithm retrieves historical correction consumption for a service point, applies a TOU map to the consumption, and writes the results to a flat file.
- **Extract Tab-Delimited Historical Corrections and Apply TOU Map (with UTC Offset)** (D2-IDEXTITUU): This algorithm retrieves historical correction consumption for a service point, applies a TOU map to the consumption, and writes the results to a tab-delimited flat file. This algorithm also supports adjusting the consumption based on Daylight Saving Time.

Batch Controls

The consumption extraction process uses a set of base package batch controls to extract and format the interval data for export.

- **Create Daily Consumption Extract Requests** (D2-CRERQ): This batch process scans for active Consumption Extract Types, and for each one that has Frequency of Automated Daily creates a pending request. This process should be scheduled to run daily (or at another regular interval).

The following sample batch controls are provided to extract and format interval data. Unique batch controls of each of these is required for each consumption extract type. You should create custom versions of the above batch controls for each consumption extract type in your implementation. Extract type-specific versions of these batch controls should be specified in the "Batch Control" section on the Consumption Extract Type as appropriate.

- **Initial Load/Ongoing Consumption Extract** (D2-IDEPD): This batch process extracts interval data for a specified period. This batch control uses the "Initial Load/Ongoing Extract Algorithm" defined on the Consumption Extract Type.
- **Historical Corrections Consumption Extract** (D2-IDEHC): This batch process extracts interval data for historical corrections. This batch control uses the "Historical Corrections Extract Algorithm" defined on the Consumption Extract Type.

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.

Example Setup Steps

Setting up consumption extracts involves the following steps:

1. Create Consumption Request Types: you should create a consumption request type for each unique type of request required by your implementation.
2. Create Consumption Extract Types: you should create a consumption extract type for each unique combination of output details (target UOM, target interval size, TOU map/template) and measurement selection criteria.
3. Add the "Create General Process Record if IMD is Historical Correction" historical correction algorithm as an Enter algorithm on the Finalized state of the initial measurement business objects.
4. Add the "Create General Process Record for Re-derived Values" historical correction algorithm as an Enter algorithm on the Finalized state on the measurement business object.
5. Create and submit initial load consumption extract requests for each consumption extract type you created earlier.
6. Set up batch processes for daily extract requests and historical corrections. Batch processing for consumption extracts should include the following:
    a. The **Create Daily Consumption Extract Requests** (D2-CRERQ) batch control should be configured to run on a regular (i.e. daily) schedule to create ongoing consumption extract requests.
    b. The **Request Monitor (Deferred)** (F1-SUBRQ) batch control should be used to monitor for pending requests and transition them to the "Submit Job" state.
    c. Historical corrections consumption extract batch controls (based on D2-IDEH, one for each consumption extract type), should be configured to run on a regular basis to check for and create historical correction extracts.

Master Data Extract

Extraction of service point, device, and install event master data is performed through use of data synchronization, audit algorithms, business objects and related batch processes.

Batch processes are used to create initial load extracts for service point, device, and install event master data. Following the initial load, data synchronization requests are created when master data is changed.

Maintenance Object Audit Algorithms

Detecting changes in service point, device, and install event master data can be detected via audit algorithms on the Service Point (D1-SP), Device Configuration (D1-DVCCONFIG), Device (D1-DEVICE),and Install Event (D1-INSTLEVT) maintenance objects. The Generic Change Data Capture (F1-GCHG-CDCP) algorithm type provided in the base package can be used to create audit algorithms for the Service Point and Install Event maintenance objects.

In addition, the "Device Change Data Capture (Install Event-Based)" (D1-IEDV-CDCP) algorithm type provided in the base package can be used to create an audit algorithm on the Device maintenance object to determine if an install event sync request record is to be created. A change in a related device's details will instantiate an Install Event Extract Sync Request (of the type defined for the "Install Event Sync Request BO" algorithm parameter) if one does not already exist in the initial state for the Install Event.

Business Objects and Algorithms

The maintenance object audit algorithms create data synchronization requests based on the "Sync Request BO" maintenance object options. Extraction of service points and install events are supported by the following data synchronization business objects.

- **SP Sync for DataConnect** (D1-ExternalRepositorySPSync): used to extract service point information. This business object should be defined as a value for the "Sync Request BO" options on the Service Point maintenance object.
- **SP/Meter Sync for DataConnect v1+**(D1-ExternalRepositorySPMtrSync): used to extract service point and meter information. This business object should be defined as a value for the "Sync Request BO" options on the Service Point, Device Configuration, and Device maintenance objects.
- **Install Event Sync for DataConnect** (D1-ExternalRepositoryIESync): used to extract install event information. This business object should be defined as a value for the "Sync Request BO" options on the Install Event 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 SP Initial Snapshot for DataConnect** (D1-SPEINISNP)
- **Capture SP/Meter Initial Snapshot for DataConnect** (D1-SPMINISNP)
- **Capture Install Event Initial Snapshot for DataConnect** (D1-IEEINISNP)

The Sync Request Monitor batch control (F1-SYNRQ) monitors synchronization requests in the Pending state and executes Monitor algorithms that check for related synchronization requests, and 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 to extract and export.

- **Capture SP Final Snapshot for DataConnect** (D1-SPEFNISNP)
- **Capture SP/Meter Final Snapshot for DataConnect v1+** (D1-SPMFINSNP)
- **Capture Install Event Final Snapshot for DataConnect** (D1-IEEFNISNP)

The "Prepare Delimited Extract Data" Enter algorithm (D1-PRPEXTDTA) on the "Send Request" state prepares 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). Note that this algorithm is defined on the "Send Request" state of the Generic Sync for DataConnect (D1-ParentExternalRepositorySyn) parent business object.

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:

- **SP Initial Load for DataConnect** (D1-SPEIL): This batch control creates initial synchronization requests for service points.
- **SP/Meter Initial Load for DataConnect v1+** (D1-SPMIL): This batch control creates initial synchronization requests for service points, device configurations, and devices.
- **Install Event Initial Load for DataConnect** (D1-IEEIL): This batch control creates initial synchronization requests for install events.

These batch processes should be run to create initial load data synchronization requests based on the "Sync for DataConnect" business objects. The following batch processes are used to create extract files from synchronization requests.

- **SP Extract for DataConnect** (D1-SPESR): This batch control creates extract file(s) that contain service point information.
- **SP/Meter Extract for DataConnect** (D1-SPMSR): This batch control creates extract file(s) that contain service point, device configuration, and device information.
- **Install Event Extract for DataConnect** (D1-IEESR): This batch control creates extract file(s) that contain install event 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 a delimited flat file containing extracted data.

Example Setup Steps

Setting up master data extracts involves the following steps:

1. Add audit algorithms on the Service Point and Install Event maintenance objects.
2. Add the DataConnect synchronization request business objects as "Sync Request BO" options on the Service Point and Install Event maintenance objects.
3. Add the "Install Event's Device Change Data Capture" algorithm as an audit algorithm on the Device maintenance object.
4. Execute initial load batch processes for service points and install events.
5. Once the initial load synchronization has been executed, changes to service points, install events, or related devices will trigger the creation of new synchronization requests and resulting extraction files.

Extract Flat File Formats

The data included in each file is based on a specific Data Area. For each Data Area, this section provides a table that lists the name and corresponding schema element and metadata field for each data element extracted by the Data Area. Reference the Data Areas below to see the details for the fields included in each output.

Process	Description	Data Area
Consumption Extract Snapshot	The Consumption Extract Snapshot data area is used for interval consumption extract.	D2-IntervalDataExtRepoSnapshot
Consumption Extract Snapshot with TOU Mapping	The Consumption Extract Snapshot (TOU Mapping) data area is used for interval consumption extract mapped to TOU periods.	D2-IntervalDataExtRepoTOUSnap
Service Point Snapshot for DataConnect	The Service Point Snapshot for DataConnect data area is used for service point master data extract.	D1-ExternalRepositorySPSnapsht
SP Meter Snapshot for DataConnect v1+	The Service Point and Meter Snapshot for DataConnect v1+ data area is used for service point and meter master data extract.	D1-ExternalRepositorySPMtrSnap
Install Event Snapshot for DataConnect	The Install Event Snapshot for DataConnect data area is used for install event master data extract.	D1-ExternalRepositoryIESnapsht

Specifics for how the flat files are created are defined on batch controls and algorithms used by the extract process.

- **File Name and Path**: Parameters on batch controls define the file name, path, and other details about the output file.
- **File Size and Contents**: Batch controls for initial load/ongoing consumption requests include a parameter to specify the number of service points to be included in each file.
- **Character Encoding and File Delimiter**: Parameters on the Prepare Delimited Extract Data (D1-PRPEXTDTA) algorithm are used to specify the character encoding and delimiter used in the flat files for master data extracts.

Example Extract File

The following example illustrates comma-separated interval data extracts based on the Consumption Extract Snapshot data area:

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366,2015-01-01,07.00.00,No

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366,2015-01-01,08.00.0,No

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366.2015-01-01,09.00.0,No

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366,2015-01-01,10.00.00,No

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366,2015-01-01,11.00.00,No

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366,2015-01-01,12.00.00,No

19502793-60E-KMUS,714532246966,19502793-60E-KMSP,KWH, ,3,600,1.366,2015-01-01,13.00.00,No

Message Formats

Oracle Utilities Meter Data Management supports three core transactional data imports: Initial Measurement Data, Device Event Data, and Usage Transactions. To see the appropriate format for each of these imports please see the following:

- **Initial Measurement Data**: Please see the IMD Seeder (D1-IMDSeeder) business object for the appropriate format
- **Device Events**: Please see the Device Event Seeder (D1-DeviceEventSeeder) business object for the appropriate format
- **Usage Transactions**: Please see the Usage Transaction Seeder (D2-UsgTranSeeder) business object for the appropriate format*

Data Import - Message Driven Bean Configuration

Overview

This section describes the steps for configuring the Message Driven Bean (MDB) feature of Oracle Utilities Meter Data Management and Oracle Utilities Smart Grid Gateway to listen to inbound JMS messages. This feature is used when importing IMDs and device events from head-end systems or when receiving Usage Transactions from CIS.

JMS Configuration

JMS configuration involves setting up JMS queues which will received inbound usage readings and device events. The JMS queues need to be created first on the application server where the OSB component is deployed. This server is referred to as remote server in the sections below. In the following section the JMS queue on the remote server is assumed to be created with the name **DestinationQueueWatch-CM**.

Note: The JMS changes described in the following sections are not persistent during patches or upgrades. They will need to be re-created after applying any patches or upgrades to Oracle Utilities Smart Grid Gateway. It is recommended to keep a backup of the `$SPLEBASE/splapp/config.xml` file.

Create a new JMS Module

Log in to the Oracle Utilities Smart Grid Gateway Weblogic console and create a JMS Module with an appropriate name. Specify the following values for this JMS module:

- **Name**: the name of JMS module. For example, JMSModule-CM
- **Target**: the name of the target server where the Oracle Utilities Smart Grid Gateway application is running. This should be specified as myserver.

Create a Foreign JMS Server

Create a Foreign JMS server under the JMS module created in the above step. Specify the following values for this foreign JMS server:

- **Name**: Name of the foreign server. For example, JMSFAServer-CM

- **Target**: This should be specified as myserver
- **JNDI Initial Context Factory**: This should be specified as weblogic.jndi.WLInitialContextFactory
- **JNDI Connection URL**: The URL of the server where OSB is deployed. For example: t3://osbserver:7001
- **JNDI Properties Credential**: Password for the OSB server user
- **JNDI Properties**: The java.naming.security.principal additional property should be specified and set to the OSB server user. For example, java.naming.security.principal=weblogic

Create a Foreign Destination

Create a Foreign destination for each remote queue. Specify the following values for this foreign destination:

- **Name**: Name of foreign destination. For instance, DestinationQueue-CM
- **Local JNDI Name**: Local JNDI name for the foreign JMS queue. For example, ForeignDestinationQueue-CM
- **Remote JNDI Name**: JNDI name of the queue on the remote server. For example, DestinationQueueWatch-CM

Create a Remote Connection Factory

Create a remote connection factory for the foreign JMS server. Specify the following values for this remote connection factory:

- **Name**: Name of remote connection factory. For example, DestinationQueueConnectionFactory-CM
- **Local JNDI Name**: Local JNDI name for the Remote Connection Factory. For example, ForegnDestinationQueueConnectionFactory-CM
- **Remote JNDI Name**: JNDI name of the JMS Connection Factory on the remote server. For example, weblogic.jms.XAConnectionFactory

Message Driven Bean Configuration

Configuration of message driven beans (MDB) involved modifying the **ejb-jar.xml** and **weblogic-ejb-jar.xml** configuration files delivered with Oracle Utilities Smart Grid Gateway. It is recommended that instead of modifying these files directly you create "Customer Modification" (CM) versions of these files to make changes to these configuration files. This ensures that your modifications are not overwritten by future application patches.

The following section describes the changes required in the CM files for configuring the MDBs to read from the foreign JMS queues set up in the steps above. This requires creating the following files under $SPLEBASE/templates:

- cm_ejb-jar.xml.wls.jms_1.include
- cm_ejb-jar.xml.wls.jms_2.include
- cm_weblogic-ejb-jar.xml.jms.include.

Note: After making these changes the initialSetup script needs to be run and Oracle Utilities Smart Grid Gateway application needs to be redeployed. However the initialSetup script will overwrite the JMS configuration changes made in the steps above. So it is recommended to keep a backup of the $SPLEBASE/splapp/config.xml file before running this script.

Changes to cm_ejb-jar.xml.wls.jms_1.include

The following is an an example of the `cm_ejb-jar.xml.wls.jms_1.include` file:

```
<message-driven>
   <description>MDB for DestinationQueue-CM</description>
   <display-name>DestinationQueueWatcher-CM</display-name>

      <ejb-name>DestinationQueueWatch-CM</ejb-name>
```

```
   <ejb-class>com.splwg.ejb.mdb.MessageProcessor</ejb-class>
   <messaging-type>javax.jms.MessageListener</messaging-type>
   <transaction-type>Bean</transaction-type>
   <message-destination-type>javax.jms.Queue</message-destination-type>
</message-driven>
```

The values specified in the above file include the following:

- **ejb-name**: This is the name of the MDB.

Changes to cm_ejb-jar.xml.wls.jms_2.include

The following is an example of the `cm_ejb-jar.xml.wls.jms_2.include` file:

```
   <assembly-descriptor>
   <security-role>
   <role-name>cisusers</role-name>
   </security-role>
   <container-transaction>
   <method>

      <ejb-name>DestinationQueueWatch-CM</ejb-name>

   <method-name>onMessage</method-name>
   </method>
   <trans-attribute>NotSupported</trans-attribute>
   </container-transaction>
   </assembly-descriptor>
```

The values specified in the above file include the following:

- **ejb-name**: This is the name of the MDB

Changes to cm_ejb-jar.xml.wls.jms_2.include

The following is an example of the `cm_weblogic-ejb-jar.xml.jms.include` file:

```
   <weblogic-enterprise-bean>

<ejb-name>DestinationQueueWatch-CM</ejb-name>

   <message-driven-descriptor>
   <pool>
   <max-beans-in-free-pool>5</max-beans-in-free-pool>
   <initial-beans-in-free-pool>1</initial-beans-in-free-pool>
   </pool>

<destination-jndi-name>ForeignDestinationQueue-CM</destination-jndi-name>

<connection-factory-jndi-name>ForeignConnectionFactory-CM</connection- factory-jndi-name>

   </message-driven-descriptor>
   </weblogic-enterprise-bean>
```

The values specified in the above file include the following:

- **ejb-name**: This should be the name of the MDB as specified in ejb-jar.xml
- **destination-jndi-name**: This should be the JNDI name of the foreign destination as provided in JMS module ' Foreign server ' Foreign destination ' Local JNDI name.
- **connection-factory-jndi-name**: This should be the JNDI name of the connection factory as provided in JMS module ' Foreign server ' Remote Connection Factory ' Local JNDI name.

Notification Queue Configuration

Payload statistics and payload summary records must be submitted sequentially in order for them to be processed correctly. To prevent them from being processed at the same time, you should set the number of notification queue polling threads to 1. Follow these steps to configure the number of notification queue threads:

1. Log in to the WebLogic Server Administration Console.
2. Under Helpful Tools, click **Configure Applications.**
3. Click on **SPLService.**
4. Click on the NotificationQueue link. for the EJB that you want to configure.
5. Go to the Configuration tab.
6. In the Change Center, click **Lock & Edit.**
7. Specify the new value of polling threads in **Max Beans in Free Pool.**
8. Click **Save.**
9. Click **Release Configuration.**
10. Restart the OUAF WebLogic instance.

Creating Application Links in External Applications

When integrating Oracle Utilities applications with other external applications, you can create links from external applications to specific portals and pages within the Oracle Utilities applications (also known as "deep linking"). These links typically leverage Business Process Assistant (BPA) scripts that direct the user to the appropriate target portal or page for a specific entity such as an account or service point. Refer to Creating Application Links in External Applications in the *Application Framework Administrative User Guide* for more information about creating links.

Note: When creating application links, the use of the following parameters will help speed up launching the target page

- **initNav=false**: skips launching the default home page for the user
- **minDashboard=true** minimizes the dashboard for the user

Go to Portal Script

The Framework provides a generic script (F1-GoToPrtl) that can be used to navigate to the maintenance portal for a specified maintenance object and entity. Refer to Navigating to a Given Record's Maintenance Portal in the *Application Framework Administrative User Guide* for more information about using this script.

Go To MDM Script

Oracle Utilities Meter Data Management includes the Go To MDM script (D1-GoToMDM) that can be used to navigate directly to a specific 360 Degree View tab. This script uses the following syntax:

```
script=D1-GoToMDM&FLAG=<FLAG_VALUE>&PKVALUE1=<ID>
```

where:

- **<FLAG_VALUE>**: A value from the DRILL_ENTITY_FLG lookup that specifies which 360 Degree View page to open.
- **<ID>**: The primary key value for the entity (such as a Device, Measuring Component, or Service Point) you wish to view.

Examples:

Script	Flag Value	ID	Example
D1-GotoMDM	FLAG=D1DV (360 View for Device)	Device ID	script=D1-GotoMDM&FLAG=D1DV&PKVALUE1={your Device ID}
D1-GotoMDM	FLAG=D1MC (360 View for Measuring Component)	Measuring Component ID	script=D1-GotoMDM&FLAG=D1MC&PKVALUE1={your Measuring Component ID}
D1-GotoMDM	FLAG=D1SP (360 View for Service Point)	Service Point ID	script=D1-GotoMDM&FLAG=D1SP&PKVALUE1={your Service Point ID}

Full URL Example:

```
https://<ENV_BASE_URL>/ouaf/cis.jsp/?minDashboard=true&initNav=false&script=D1-
GotoMDM&FLAG=D1DV&PKVALUE1={your Device ID}
```

Chapter 21

Data Conversion and Migration

This chapter provides guidelines for migrating and/or converting data between application environments, including moving data from existing applications into Oracle Utilities Customer Cloud Service or Oracle Utilities Meter Solution Cloud Service. Existing applications can include legacy applications as well as on-premises implementations of Oracle Utilities applications such as Oracle Utilities Customer Care and Billing and Oracle Utilities Meter Data Management.

This document provides guidelines for migrating and/or converting specific types of data in a number of specific application configurations (see Scenarios, below).

See Conversion in the *Application Framework Administrative User Guide* for information about the general conversion process.

This chapter also provide information about migrating master and transaction data between environments for testing purposes.

Terms and Definitions

The process described in this chapter uses the following terms:

Term	Definition
Synchronize	Data is created through a data synchronization process. For example, Usage Subscriptions used by Meter Data Management functionality are synchronized from Service Agreements used by Customer Care and Billing functionality. See Data Synchronization in the *Application Framework Administrative User Guide* for information about the data synchronization process.
Upload	Data is uploaded directly into the database from a legacy application.
Bind	Direct relationships between records, such as between devices and assets, are created via batch processing. See **Binding MDM Device ID and ODM Asset ID** for more information.
Import	Data is imported from an existing Oracle Utilities application such as Oracle Utilities Customer Care and Billing and Oracle Utilities Meter Data Management.

Term	Definition
Legacy CIS	A legacy customer information system from which data will be migrated/converted.
Legacy MDM	A legacy meter data management application from which data will be migrated/converted.
CCB (CC&B)	Oracle Utilities Customer Care and Billing
MDM	Oracle Utilities Meter Data Management
ODM	Oracle Utilities Operational Device Management

Database Tables

The process described in this chapter references a number of different types of database tables. These are described below.

Production Tables: Tables used by application when running in a production environment. Examples of production tables include CI_SP (Service Point - CCB), D1_SP (Service Point - MDM), W1_ASSET (Asset). Production tables are accessible through the **Table** portal in the Customer Cloud Service application.

Staging Tables: Tables used to facilitate import and migration into the product database. Staging tables are not accessible through the Customer Cloud Service application user interface.

Key Tables: Tables used to facilitate generation of keys. Examples of key tables include CI_SP_K (Service Point Key), D1_SP_K (Service Point - MDM), W1_ASSET_K (Asset Key). Key tables are accessible through the **Table** portal in the Customer Cloud Service application.

Scope and Assumptions

The process described in this chapter is based on the following scope and assumptions:

Legacy data has been loaded to the staging tables. See **Data Conversion Support for Cloud Implementations** in the *Oracle Utilities Cloud Service Foundation Administrative User Guide* for information about loading data into staging tables.

The process described in this document covers Meter Data Management conversion delivered for converting legacy master data such as Contacts Devices, Device Configurations, Measuring Components, Service Points, Install Events, and Usage Subscriptions.

The process described in this document includes Operational Device Management conversion pertinent to deployments of Oracle Utilities Customer Cloud Service that include customer, meter and operational device (asset) functionality.

Additional Information

Refer to the following documentation for additional information about data conversion and migration with Oracle Utilities cloud services:

- *Cloud Service Foundation Administrative User Guide*
- *Data Upload Support Guide*

These documents can be found in the **Supporting Cloud Service Guides** section on the Oracle Utilities Customer Cloud Service or Oracle Utilities Meter Solution Cloud Service documentation website.

Data Migration Scenarios

This section outlines the ways in which specific types of data are migrated/converted in a number of specific application configurations.

The following abbreviations are used in the scenarios described in this section.

Abbreviation	Description
C2MO	Customer Cloud Service including Customer Care and Billing, Meter Data Management, and Operational Device Management functionality.
C2M	Customer Cloud Service including Customer Care and Billing and Meter Data Management functionality.
MDM/ODM	Meter Solution Cloud Service including Meter Data Management and Operational Device Management functionality.

Notes:

- The tables below layout the data in each of the functional areas (Customer Care and Billing, Meter Data Management, and Operational Device Management) as appropriate.
- Data of corresponding types are aligned in the same row in each table. For example, Person data in CCB corresponds to Contact data in MDM.

Legacy Customer Information System to C2MO

Customer Care and Billing Data		Meter Data Management Data		Operational Device Data	
Person	Upload from Legacy CIS	Contact	Synchronize from CCB		
Account	Upload from Legacy CIS				
Service Agreement (SA)	Upload from Legacy CIS	Usage Subscription (US)	Synchronize from CCB		
Premise	Upload from Legacy CIS				
Service Point (SP)	Upload from Legacy CIS	Service Point (SP)	Synchronize from CCB	Node	Synchronize from MDM
		Device	Upload from Legacy CIS Bind Device IDs	Asset	Upload from Legacy CIS Bind Device IDs
		Device Configuration	Upload from Legacy CIS		
		Measuring Component	Upload from Legacy CIS		
		Install Event	Upload from Legacy CIS Has Foreign Keys to Service Point that do not exist in staging because they were synchronized.	Asset Node	Synchronize from MDM

Legacy Customer Information System to C2M

Customer Care and Billing Data		Meter Data Management Data	
Person	Upload from Legacy CIS	Contact	Synchronize from CCB

Customer Care and Billing Data		Meter Data Management Data	
Account	Upload from Legacy CIS		
Service Agreement (SA)	Upload from Legacy CIS	Usage Subscription (US)	Synchronize from CCB
Premise	Upload from Legacy CIS		
Service Point (SP)	Upload from Legacy CIS	Service Point (SP)	Synchronize from CCB
		Device	Upload from Legacy CIS
		Device Configuration	Upload from Legacy CIS
		Measuring Component	Upload from Legacy CIS
		Install Event	Upload from Legacy CIS

Legacy Meter Data Management to MDM/ODM

Meter Data Management Data		Operational Device Management Data	
Contact	Upload from Legacy MDM		
Usage Subscription (US)	Upload from Legacy MDM		
Service Point (SP)	Upload from Legacy MDM	Node	Synchronize from MDM
Device	Upload from Legacy MDM	Asset	Upload from Legacy MDM
	Bind Device IDs		Bind Device IDs
Install Event	Upload from Legacy MDM	Asset Node	Synchronize from MDM

Customer Care and Billing to C2M

Customer Care and Billing Data		Meter Data Management Data	
Person	Import from CCB	Contact	Synchronize from CCB
Account	Import from CCB		
Service Agreement (SA)	Import from CCB	Usage Subscription (US)	Synchronize from CCB
Premise	Import from CCB		
Service Point (SP)	Import from CCB	Service Point (SP)	Synchronize from CCB
		Device	Upload from CCB
		Device Configuration	Upload from CCB
		Measuring Component	Upload from CCB
		Install Event	Upload from CCB

Customer Care and Billing to C2MO

Customer Care and Billing Data		Meter Data Management Data		Operational Device Management Data
Person	Import from CCB	Contact	Synchronize from CCB	
Account	Import from CCB			

Customer Care and Billing Data		Meter Data Management Data		Operational Device Management Data	
Service Agreement (SA)	Import from CCB	Usage Subscription (US)	Synchronize from CCB		
Premise	Import from CCB				
Service Point (SP)	Import from CCB	Service Point (SP)	Synchronize from CCB	Node	Synchronize from MDM
		Device	Upload from CCB	Asset	Upload from CCB
			Bind Device IDs		Bind Device IDs
		Device	Upload from CCB		
		Measuring Component	Upload from CCB		
		Install Event	Upload from CCB	Asset Node	Synchronize from MDM

Customer Care & Billing + Meter Data Management to C2M

Customer Care and Billing Data		Meter Data Management Data	
Person	Import from CCB	Contact	Import from MDM
Account	Import from CCB		
Service Agreement (SA)	Import from CCB	Usage Subscription (US)	Import from MDM
Premise	Import from CCB		
Service Point (SP)	Import from CCB	Service Point (SP)	Import from MDM
		Device	Import from MDM
		Device Configuration	Import from MDM
		Measuring Component	Import from MDM
		Install Event	Import from MDM

Customer Care & Billing + Meter Data Management + Operational Device Management to C2MO

Customer Care and Billing Data		Meter Data Management Data		Operational Device Management Data	
Person	Import from CCB	Contact	Import from MDM		
Account	Import from CCB				
Service Agreement (SA)	Import from CCB	Usage Subscription (US)	Import from MDM		
Premise	Import from CCB				
Service Point (SP)	Import from CCB	Service Point (SP)	Import from MDM	Node	Import from ODM
		Device	Import from MDM	Asset	Import from ODM
		Device Configuration	Import from MDM		
		Measuring Component	Import from MDM		

Customer Care and Billing Data		Meter Data Management Data		Operational Device Management Data	
		Install Event	Import from MDM	Asset Node	Import from ODM

Customer Care & Billing + Meter Data Management to C2MO

Customer Care and Billing Data		Meter Data Management Data		Operational Device Management Data	
Person	Import from CCB	Contact	Import from MDM		
Account	Import from CCB				
Service Agreement (SA)	Import from CCB	Usage Subscription (US)	Import from MDM		
Premise	Import from CCB				
Service Point (SP)	Import from CCB	Service Point (SP)	Import from MDM	Node	Synchronize from MDM
		Device	Import from MDM	Asset	Upload from MDM
		Device Configuration	Import from MDM		
		Measuring Component	Import from MDM		
		Install Event	Import from MDM	Asset Node	Synchronize from MDM

Meter Data Management to MDM/ODM

Meter Data Management Data		Operational Device Management Data	
Contact	Import from MDM		
Usage Subscription (US)	Import from MDM		
Service Point (SP)	Import from MDM	Node	Synchronize from MDM
Device	Import from MDM	Asset	Upload from Legacy MDM
Device Configuration	Import from MDM		
Measuring Component	Import from MDM		
Install Event	Import from MDM	Asset Node	Synchronize from MDM

Data Migration Processes

This section provides specifics regarding the processes used when migrating/converting customer data, meter data, and measurement data.

Customer Data Migration

This section outlines the steps for migrating customer data to an Oracle Utilities Customer Cloud Service implementation that includes Customer Care and Billing, Meter Data Management, and Operational Device Management functionality.

Legacy Customer Information System Upgrading to Oracle Utilities Cloud Service

In this scenario a new customer using a legacy CIS application is upgrading to Oracle Utilities Customer Cloud Service. This outline assumes that the legacy data has been loaded into staging tables.

Step	Description	Remarks
1	Convert legacy person, account, premise, service point and service agreement to CC&B person, account, premise, service point and service agreement.	
2	Convert legacy meter, meter configuration, register to MDM device, device configuration and measuring component.	See Meter Conversion.
3	Synchronize CC&B person, service point and service agreement to MDM contact, service point and usage subscription.	
4	Copy production MDM SP to staging MDM SP	See Meter Data Management Install Event Migration.
5	Build the staging MDM SP key mapping table.	
6	Convert legacy install event to MDM install event.	
7	Convert legacy meter to ODM asset.	The following steps only apply if ODM functionality is included.
8	Bind the MDM device to ODM asset.	See Binding Meter Data Management Device IDs and Operational Device Management Asset IDs.
9	Bind the ODM asset to MDM device.	See Binding Meter Data Management Device IDs and Operational Device Management Asset IDs.
10	Synchronize MDM service point to ODM service point.	
11	Synchronize MDM install event to ODM asset disposition	

Customer Care and Billing Upgrading to Oracle Utilities Cloud Service

In this scenario an existing CC&B customer is upgrading to Oracle Utilities Customer Cloud Service. This outline assumes that the CC&B meter data has been loaded into the MDM and ODM meter and asset staging tables.

Step	Description	Remarks
1	Export CC&B database to single instance database.	

Step	Description	Remarks
2	Convert CC&B meter, meter configuration and register to MDM device, device configuration and measuring component.	See Meter Conversion.
3	Synchronize CC&B person, service point and service agreement to MDM contact, service point and usage subscription.	
4	Build the staging CC&B SP key mapping table.	See Meter Data Management Install Event Migration.
5	Copy production MDM SP to staging MDM SP.	
6	Build the staging MDM SP key mapping table.	
7	Convert CC&B install events to MDM install events.	
8	Convert CC&B meters to ODM assets.	The following steps only apply if ODM functionality is included.
9	Bind the MDM devices to ODM assets.	See Binding Meter Data Management Device IDs and Operational Device Management Asset IDs.
10	Bind the ODM assets to MDM devices.	See Binding Meter Data Management Device IDs and Operational Device Management Asset IDs.
11	Synchronize MDM service points to ODM service points.	
12	Synchronize MDM install events to ODM asset dispositions.	

Integrated Customer Care & Billing and Meter Data Management Upgrading to Oracle Utilities Cloud Service

In this scenario an existing customer using integrated CC&B and MDM applications is upgrading to Oracle Utilities Customer Cloud Service. This outline assumes that the MDM meter data has been loaded into the ODM staging tables.

Step	Description	Remarks
1	Export CC&B database to database.	
2	Export MDM database to database.	
3	Convert MDM meters to ODM assets.	The following steps only apply if ODM functionality is included.
4	Synchronize MDM service points to ODM service points.	
5	Synchronize MDM install events to ODM asset dispositions.	
6	Create the MDM device identifiers from ODM asset identifiers.	

Integrated Customer Care & Billing and Meter Data Management Upgrading to Oracle Utilities Cloud Service

In this scenario an existing customer using integrated CC&B and MDM applications is upgrading to Oracle Utilities Customer Cloud Service. This outline assumes that the MDM meter data has been loaded into the ODM staging tables.

Step	Description	Remarks
1	Export CC&B database to database.	
2	Export MDM database to database.	
3	Convert MDM meters to ODM assets.	The following steps only apply if ODM functionality is included.
4	Synchronize MDM service points to ODM service points.	
5	Synchronize MDM install events to ODM asset dispositions.	
6	Create the MDM device identifiers from ODM asset identifiers.	

Integrated Customer Care & Billing, Meter Data Management and Operational Device Management Upgrading to Oracle Utilities Cloud Service

In this scenario an existing customer using integrated CC&B, MDM and ODM applications is upgrading to Oracle Utilities Customer Cloud Service.

Step	Description	Remarks
1	Export CC&B database to database.	
2	Export MDM database to database.	
3	Export ODM database to database.	

Meter Data Management Install Event Migration

This section describes the issue when converting Meter Data Management install events into Oracle Utilities Customer Cloud Service using master data synchronization and other batch processes.

Install Event Conversion from Legacy CIS application

When migrating legacy data from a CIS application to Oracle Utilities Customer Cloud Service implementation that includes Customer Care and Billing, Meter Data Management, and Operational Device Management functionality, the approach is to stage the customer related legacy data to corresponding CC&B staging tables.

After customer-related data such as persons, service points, and service agreements are successfully converted in the CC&B production tables, the initial load batch jobs are submitted to synchronize this customer-related data to MDM as contacts, service points, and usage subscriptions.

Similarly, after meter-related data such as devices, device configurations, measuring components, and install events are loaded to corresponding MDM staging tables, conversion batch jobs should be submitted.

The conversion process works well for devices, device configurations, and measuring components but not for install events.

This is due to the fact that MDM service point production table is populated through a synchronization process which results in an empty staging MDM SP table (D1_SP) and MDM SP legacy and production key mapping table (DK_SP), which are used in the foreign key resolution when inserting records to production MDM SP table.

To fix this issue, the staging MDM SP table (D1_SP) and MDM SP legacy and production key mapping table (DK_SP) must be populated with data from production MDM SP and SP identifier tables.

The following batch jobs are required to be run to populate these tables before running the MDM install event conversion process.

Batch Control	Description	Remarks
X1CPSTSP	Copy To Staging D1_SP	Populates the staging MDM SP be copying the data from production SP with the legacy SP ID as the primary key
X1PODSPK	Copy To Staging DK_SP	Populates the MDM SP legacy and production key mapping table with the old key as legacy SP ID and with the new key as production SP ID

Install Event Conversion from Customer Care and Billing

When migrating legacy data from CC&B to Oracle Utilities Customer Cloud Service implementation that includes Customer Care and Billing, Meter Data Management, and Operational Device Management functionality, the approach is to re-use the existing customer-related data by porting these to the database using export/import or data pump database functions.

Customer-related data such as persons, service points, and service agreements are synchronized to MDM contacts, service points, and usage subscriptions through initial load batch jobs.

CC&B meter-related data such as meters, meter configurations, registers, and install events are loaded to corresponding MDM staging tables (devices, device configurations, measuring components, and install events) and conversion batch jobs should be submitted.

The same issue will be encountered regarding the install events conversion due to the empty staging MDM SP table (D1_SP) and MDM SP legacy and production key mapping table (DK_SP), used in the foreign key resolution when inserting records to production MDM SP table. In addition, the job that populates the MDM SP legacy and production key mapping table (DK_SP) will not work because it is dependent on the CC&B SP legacy and production key mapping table (CK_SP).

To fix this issue, the CC&B SP legacy and production key mapping table (CK_SP) will have to be populated with data from production CC&B SP.

The following batch jobs are required to be run to populate these tables before running the MDM install event conversion process.

Batch Control	Description	Remarks
X1POCSPK	Copy To Staging CK_SP	Populates the CC&B legacy and production key mapping table old and new keys with the production CC&B SP ID
X1CPSTSP	Copy To Staging D1_SP	Populates the staging MDM SP be copying the data from production SP with the legacy SP ID as the primary key
X1PODSPK	Copy To Staging DK_SP	Populates the MDM SP legacy and production key mapping table with the old key as legacy SP ID and with the new key as production SP ID

Meter Data Migration

This section outlines the steps for migrating customer data to an Oracle Utilities Meter Solution Cloud Service implementation that includes Meter Data Management, and Operational Device Management functionality.

Legacy Meter Data Management Upgrading to Oracle Utilities Meter Solution Cloud Service

In this scenario a new customer using a legacy MDM application is upgrading to Oracle Utilities Meter Solution Cloud Service. This outline assumes that the legacy data has been loaded into the staging tables.

Step	Description	Remarks
1	Convert legacy contacts, service points, and usage subscriptions to MDM contacts, service points, and usage subscriptions.	See Meter Conversion.
2	Convert legacy devices, device configurations, measuring components to MDM devices, device configurations, and measuring components.	See Meter Conversion.
3	Convert legacy install events to MDM install events.	See Meter Conversion.
4	Convert legacy meters to ODM assets.	The following steps only apply if ODM functionality is included.
5	Bind the MDM devices to ODM assets.	See Binding Meter Data Management Device IDs and Operational Device Management Asset IDs.
6	Bind the ODM assets to MDM devices.	See Binding Meter Data Management Device IDs and Operational Device Management Asset IDs.
7	Synchronize MDM service points to ODM service points.	
8	Synchronize MDM install events to ODM asset dispositions.	

Meter Data Management Upgrading to Oracle Utilities Meter Solution Cloud Service

In this scenario an existing customer using the MDM application is upgrading to Oracle Utilities Meter Solution Cloud Service. This outline assumes that the MDM meter has been loaded into the ODM asset staging tables.

Step	Description	Remarks
1	Export MDM database to the database.	
2	Convert MDM meters to ODM assets.	The following steps only apply if ODM functionality is included.
3	Create MDM device identifiers from ODM asset identifiers.	
4	Synchronize MDM service point to ODM service point.	

Step	Description	Remarks
5	Synchronize MDM install event to ODM asset disposition.	

Integrated Meter Data Management and Operational Device Management Upgrading to Oracle Utilities Meter Solution Cloud Service

In this scenario an existing customer using integrated MDM and ODM applications is upgrading to Oracle Utilities Meter Solution Cloud Service.

Step	Description	Remarks
1	Export MDM database to database.	
2	Export ODM database to database.	

Meter Conversion

The table below shows the logical sequence in running the meter conversion jobs implementing the framework conversion tools. This outline assumes that the legacy data has been loaded into the staging tables

	Contact	Device	Device Configuration	Measuring Component	Service Point	Usage Subscription	Install Event
Insertion	Insert D1_ CONTACT (D1CNT00I)	Insert D1_ DVC (D1DVC00I)	Insert D1_ DVC_CFG (D1DC000I)	Insert D1_ MEASR_ COMP (D1MC000I)	Insert D1_SP (D1SP000I)	Insert D1_US (D1US000I)	Insert D1_ INSTALL_EVT (D1IE000I)
Key Generation	Generate Keys for D1_ CONTACT (D1CNT00K)	Generate Keys for D1_DVC (D1DVC00K)	Generate Keys for D1_DVC_ CFG (D1DC000K)	Generate Keys for D1_ MEASR_ COMP (D1MC000K)	Generate Keys for D1_SP (D1SP000K)	Generate Keys for D1_US (D1US000K)	Generate Keys for D1_ INSTALL_EVT (D1IE000K)
Legacy Data Validation	Validate D1-CONTACT (D1CNT00V)	Validate D1-DEVICE (D1DVC00V)	Validate D1-DVCCONFIG (D1DC000V)	Validate D1-MEASRCOMP (D1MC000V)	Validate D1-SP (D1SP000V)	Validate D1-US (D1US000V)	Validate D1-INSTLEVT (D1IE000V)
Production Data Validation	Validate D1-CONTACT (D1CNT00V)	Validate D1-DEVICE (D1DVC00V)	Validate D1-DVCCONFIG (D1DC000V)	Validate D1-MEASRCOMP (D1MC000V)	Validate D1-SP (D1SP000V)	Validate D1-US (D1US000V)	Validate D1-INSTLEVT (D1IE000V)
XML Resolution	Resolve XML for D1-CONTACT (D1CNT00R)	Resolve XML for D1-DEVICE (D1DVC00R)	Resolve XML for D1-DVCCONFIG (D1DC000R)	Resolve XML for D1-MEASRCOMP (D1MC000R)	Resolve XML for D1-SP (D1SP000R)	Resolve XML for D1-US (D1US000R)	Resolve XML for D1-INSTLEVT (D1IE000R)
	Insert D1_ CONTACT_ IDENTIFIER (D1CNTIDI)	Insert D1_ DVC_ IDENTIFIER (D1DVCIDI)	Insert D1_ DVC_CFG_ CHAR (D1DCCHRI)	Insert D1_ MEASR_ COMP_ IDENTIFIER (D1MCIDTI)	Insert D1_SP_ IDENTIFIER (D1SPIDTI)	Insert D1_US_ IDENTIFIER (D1USIDTI)	Insert D1_ INSTALL_ EVT_CHAR (D1IECHRI)

	Contact	Device	Device Configuration	Measuring Component	Service Point	Usage Subscription	Install Event
	Insert D1_ CONTACT_ CHAR (D1CNTCHI)	Insert D1_ DVC_CHAR (D1DVCCHI)		Insert D1_ MEASR_ COMP_CHAR (D1MCCHRI)	Insert D1_SP_ CHAR (D1SPCHRI)	Insert D1_US_ CHAR (D1USCHRI)	
	Insert D1_ CONTACT_ NAME (D1CNTNMI)			Insert D1_ MEASR_ COMP_REL (D1MCRELI)	Insert D1_SP_ CONTACT (D1SPCNTI)	Insert D1_US_ CONTACT (D1USCNTI)	
	Insert D1_ CONTACT_ PHONE (D1CNTPHI)				Insert D1_SP_ EQPMNT (D1SPEQPI)	Insert D1_US_ SP (D1USSP0I)	
						Insert D1_US_ SP_CHAR (D1USSPCI)	

Binding Meter Data Management Device IDs and Operational Device Management Asset IDs

This section describes the batch jobs used to populate the device identifier and asset identifier tables to bind device and asset IDs.

Device and Asset are Loaded with Converted Data

This section applies to data migration scenarios where both device and asset tables are loaded with legacy data from either a CIS application or existing CC&B application upgrading to Oracle Utilities Customer To Meter.

The following batch jobs must be run to populate the both Device Identifier and Asset Identifier tables to bind both converted device and asset IDs.

Batch Control	Description	Remarks
X1-LIDAD	Populate Asset Identifier with Device from Legacy Device	
X1-LIDDA	Populate Device Identifier with Asset from Legacy Device	

Asset Loaded with Converted Data

This section applies to the following data migration scenario:

- Existing separate instances of CC&B and MDM applications upgrading to Oracle Utilities Customer Cloud Service (or Customer To Meter)
- Existing MDM application upgrading to Oracle Utilities Meter Solution Cloud Service

The following batch job must be run to populate the Device Identifier table with an entry that binds the device ID with the converted asset ID.

Batch Control	Description	Remarks
X1-IDATD	Populate Device Identifier from Asset	

Asset Conversion

Conversion Sequence

	Node	Asset
Legacy Data Validation	Validate W1-NODE (W1NOD00V)	Entity Validation for W1-ASSET (W1AST00V)
Key Generation	Generate Keys for W1_NODE (W1NOD00K)	Generate Keys for W1_ASSET (W1AST00K)
XML Resolution	Resolve XML for W1-NODE (W1NOD00R)	Resolve XML for W1_ASSET (W1AST00R)
Insertion	Insert W1_NODE (W1NOD00I)	Insert W1_ASSET (W1AST00I)
	Insert W1_NODE_CHAR (W1NODCHI)	Insert W1_ASSET_IDENTIFIER (W1ASTIDI)
	Insert W1_NODE_IDENTIFIER (W1NODIDI)	Insert W1_ASSET_NODE (W1ASTNDI)
		Insert W1_ASSET_CHAR (W1ASTCHI)
Production Data Validation	Validate W1-NODE (W1NOD00V)	Entity Validation for W1-ASSET (W1AST00V)

Measurement Data Migration

Migrating data from the Measurement table (D1_MSRMT) requires a special treatment because of the extremely large volume of data being migrated.

For cloud service implementations, the recommendation is to directly upload the legacy measurements into the production Measurement table using the Cloud Service Foundation tool. This is because the data conversion process will be much faster compares to staging the legacy data and running conversion jobs.

Implementations may opt to use the Oracle Utilities Application Framework conversion tool to stage measurement and then mass insert the staged data into production measurement table.

Caveats with this approach:

- It will be a slower process compared to directly loading the data.
- Insertion batch controls are not provided with MDM, however, implementation should be able to create custom batch controls based on templates provided with the OUAF.

On-premises implementations can also directly upload the legacy measurements to the production measurement table. The only difference is that the Cloud Service Foundation tool is not available, so the implementation will have to use SQL loader. Alternatively, they may opt for measurement staging approach using the OUAF conversion tool.

SQL Loader offers various performance improvement measures and supports default field values, date time caching, multiple record types (delimited, fixed length, binary)

The product supports the ability to create 100% custom control file for specific table.

Required Configuration for Measurement Upload

Control File

The file for legacy measurement contains the legacy register or channel (measuring components) that need to be resolved before mass loading the data into the production database.

This process uses a specific control file to optimize the upload performance. This control file will be responsible for resolving or deriving the production measuring component key for the corresponding legacy key upon insertion to the production table.

This control file is called **D1_MSRMT_CTL.ctl**, and can be found in the **Data Upload Sample Data Files** zip file available in the **Supporting Cloud Service Guides** section on the Oracle Utilities Customer Cloud Service or Oracle Utilities Meter Solution Cloud Service documentation website.

Create Managed Content for the Control File

1. Select **Admin**, then **System**, then **Managed Content**, then **Add**.
2. Enter a code for the control file in the **Managed Content** field.
3. Select "XML" from the **Managed Content Type** drop-down list.
4. Enter a name for the control file in the **Description** field.
5. Click the **Schema** tab, and paste the D1_MSRMT_CTL.ctl control file text into the **Editor** area.
6. Click **Save**.

Create the Conversion Task Type for the Measurement Table (D1_MSRMT)

1. Select **Admin**, then **Conversion Support**, then **Conversion Task Type**, then **Add**.
2. Select “Conversion Instructions - General” from the **Service Task Type** drop-down list and click **OK**.
3. Enter a code and **Description** for the conversion task type.
4. Select “Conversion Artifacts - Table” from the **Related Transaction BO** drop-down list.
5. Select the managed content created from the previous step from the **Override Control File** drop-down list in the **Conversion Artifacts Instructions** section.
6. Click **Save**.

Setup the Data Upload Master Configuration

1. Select **Admin**, then **General**, then **Master Configuration**.
2. Select and broadcast 'Conversion Data Upload Configuration' from the list
3. Click **Edit**.
4. Add an enter in the **Override Instructions - Table** section for the Measurement table (D1_MSRMT) and specify the Conversion Task Type created from the previous step.
5. Click **Save**.

Generate Conversion Artifacts for the Measurement Table (D1_MSRMT)

1. Select **Admin**, then **Conversion Support**, then **Generate Conversion Artifacts**.

2. Enter or search for D1_MSRMT in the **Table** field.
3. Click **Generate**.
4. A warning appears, click **OK** and then click **Continue** when prompted.

Run Conversion Batch Processing for the Measurement Table (D1_MSRMT)

1. Upload the file containing legacy data to an Object Storage location. This location should be defined as a value for the File Storage Configuration (F1-FileStorage) extendable lookup.
2. Run the Conversion - Load Data using SQL Loader batch process (K1-CNVLD) using the following parameters:
   - **Input File Storage**: The Object Storage location (defined as a value for the File Storage Configuration (F1-FileStorage) extendable lookup) that contains the file to be uploaded.
   - **Table**: D1_MSRMT
   - Other parameters as appropriate.

Refer to the *Cloud Service Foundation Administrative User Guide* in the **Supporting Cloud Service Guides** section on the Oracle Utilities Customer Cloud Service or Oracle Utilities Meter Solution Cloud Service documentation website for more information about running conversion batch controls with files in Object Storage locations.

Migrating Data for On-Premises Implementations

The processes documented here can also apply to on-premises implementations. The key difference is that on-premises implementations will not be able to use Oracle Utilities Cloud Service Foundation to move data into the staging tables.

Migrating Master and Transaction Test Data

This section provides a list of Configuration Migration Assistant (CMA) Migration Plans that can be used to migrate master and transaction data related to usage subscriptions from one environment to another (it is assumed that all admin data will have already been set up in the target environment). These migration plans will allow implementations to migrate master and transaction data from a production environment to a Test or Development environment to recreate production issues and execute regression test scenarios.

The table below lists the migration plans that can be used to migrate master and transaction data. Use the **Migration Plan** portal to view details for these migration plans, including their Primary and Subordinate instructions and the business objects and Traversal Criteria used with each.

Migration Plan	Description
D1-Activity	Activity
D1-AttributeDataSnapshot	Attribute Data Snapshot
D1-CommOut	Communication Out
D1-Device	Device
D1-DeviceConfiguration	Device Configuration
D2-DynamicOption	Dynamic Option
D1-Facility	Facility
D1-InitialMeasurementData	Initial Measurement Data
D1-InstallEvent	Install Event
D1-Measurement	Measurement
D1-MeasuringComponent	Measuring Component
D1-DeviceEvent	Device Event

Migration Plan	Description
D1-ServicePoint	Service Point
F1-ServiceTask	Service Task
D2-UsageSubscription	Usage Subscription
D2-USMarketParticipant	Usage Subscription Market Participant
D2-UsageTransaction	Usage Transaction
D2-UTException	UT Exception
D1-VEEException	VEE Exception

Refer to Configuration Migration Assistant in the *Administrative User Guide* for more information about using Configuration Migration Assistant.