Oracle® Revenue Management and Billing

Version 2.8.0.0.0

Administrative Guide

Revision 2.0

F25391-01

May, 2020

ORACLE®

Contents

Notices

Oracle Revenue Management and Billing Version 2.8.0.0.0 Administrative Guide

F25391-01

Trademark Notice

Oracle, Java, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation 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.

License Restrictions Warranty/Consequential Damages Disclaimer

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 de-compilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

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.

Restricted Rights Notice

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

> **U.S. GOVERNMENT RIGHTS**
>
> Oracle programs, including any operating system, integrated software, any programs installed on the hardware, documentation, and/or technical data delivered to U.S. Government end users are “commercial computer software” or “commercial technical data” 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, documentation, and/or technical data shall be subject to license terms and restrictions as mentioned in Oracle License Agreement, and to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software 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 in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on 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.

Preface

About this Document

This document lists and describes the various feature-specific options that the system administrator or the implementation team has to configure in order to use the system.

Intended Audience

This document is intended for the following audience:

- System Administrators
- Consulting Team
- Implementation Team
- Development Team

Organization of the Document

This document contains the following chapters:

Chapter Number	Chapter Name	Chapter Description
Chapter 1	Preparing To Implement	Lists and describes how to setup various features, such as the open-item accounting, fund accounting, payment event distribution, loans, quotes, scripting, reports, cases, batch scheduler, zones, and To Dos in the system.
Chapter 2	Defining General Options Addendum	Explains how to setup the installation options, customer languages, accounting calendars, banks, and service types. It also explains how to view the To Do types and assign a To Do role to a To Do type. In addition, it explains how to view the audit trial of a table.
Chapter 3	Defining Financial Transaction Options	Describes the bills, payments, adjustments, and their financial transactions. It also explains how to create divisions, revenue classes, distribution codes, billable charge templates, bill segment types, deposit classes, payment segment types, adjustment types, adjustment type profiles, approval profiles, tender types, tender sources, auto pay sources, auto pay route types, auto pay instructions, cancel reasons, and billable charge line types. It also describes adjustment approval process and automatic payments. In addition, its explains how to setup the payment event distribution, payables cash accounting, open item accounting, and fund accounting in the system.
Chapter 4	Defining Customer Options	It describes and explains how to setup various pre-requisites which are required for creating a person, account, statement construct, customer contact, contract, and order in the system.
Chapter 5	Defining Credit & Collections Options	It describes the collection and write off processes. It also explains how to design and setup the collection and write off procedures.

Chapter Number	Chapter Name	Chapter Description
Chapter 6	Defining Cycles	It describes the bill cycles, bill periods, and statement cycles. It also explains how to setup the bill cycles, bill periods, and statement cycles in the system.
Chapter 7	Defining Contract Types	It describes how to design contract types for contracts. It also explains how to setup the contract types in the system.
Chapter 8	Background Processes Addendum	It describes and explains how to setup various processes in the background mode.
Chapter 9	Defining Service Credit Options	It describes service credit and explains how to setup various pre-requisites which are required for creating a service credit. In addition, it lists various examples in which the service credits can be earned.
Chapter 10	Defining Loan Options	It describes various processes related to loan and explains how interest calculation takes place in the system. It also explains how to enable the loans feature in the system.
Chapter 11	Defining Non-billed Budget Options	It describes non-billed budget and explains how to activate, renew, expire, and stop the non-billed budgets in the system. It also explains how to setup various pre-requisites which are required for creating a non-billed budget.
Chapter 12	Defining Quotation Options	It explains how to setup contract types for quotes, quote route types, decline reasons, and customer classes for quotes.
Chapter 13	Defining Case Management Options	It describes cases and explains how to setup various pre-requisites which are required for creating a case in the system.
Chapter 14	Workflow and Notification Options	It describes the workflow and notification process. It also explains how to setup various pre-requisites which are required for setting up the workflow and notification procedures.
Chapter 15	Security Addendum	It describes how to implement the account security. It also explains how to mask sensitive data in the system.
Chapter 16	Defining Overdue Processing Options	It describes the overdue process and how the overdue monitor works. It also explains how to setup various pre-requisites which are required for setting up the overdue procedures.
Chapter 17	Defining Batch Schedule Options	It describes how to schedule batch jobs. It also explains how to setup various pre-requisites which are required for setting up the batch scheduler.
Chapter 18	Configuring Zones	It describes how to configure timeline zones.
Chapter 19	CTI-IVR Integration	It explains how to integrate the system with the CTI-IVR.

Related Documents

You can refer to the following documents:

Document Name	Description
Oracle Revenue Management and Billing Version 2.8.0.0.0 Release Notes	Provides a brief description about the new features, enhancements, UI and database level changes, supported platforms, framework upgrade, supported upgrades, and technology upgrade made in this release. It also highlights the discontinued features, bug fixes, and known issues in this release.
Oracle Revenue Management and Billing Business Process Guide	Lists and describes various features available in the system which can be used in both the financial services and insurance domains.
Oracle Revenue Management and Billing Banking User Guide	Lists and describes various banking features in Oracle Revenue Management and Billing. It also describes all screens related to these features and explains how to perform various tasks in the application.
Oracle Revenue Management and Billing Insurance User Guide	Lists and describes various insurance features in Oracle Revenue Management and Billing. It also describes all screens related to these features and explains how to perform various tasks in the application.
Oracle Revenue Management and Billing TFM - Batch Execution Guide	Describes the sequence in which the batches must be executed while performing various tasks in the Transaction Feed Management module.
Oracle Revenue Management and Billing Batch Guide	Lists and describes various ORMB batches.

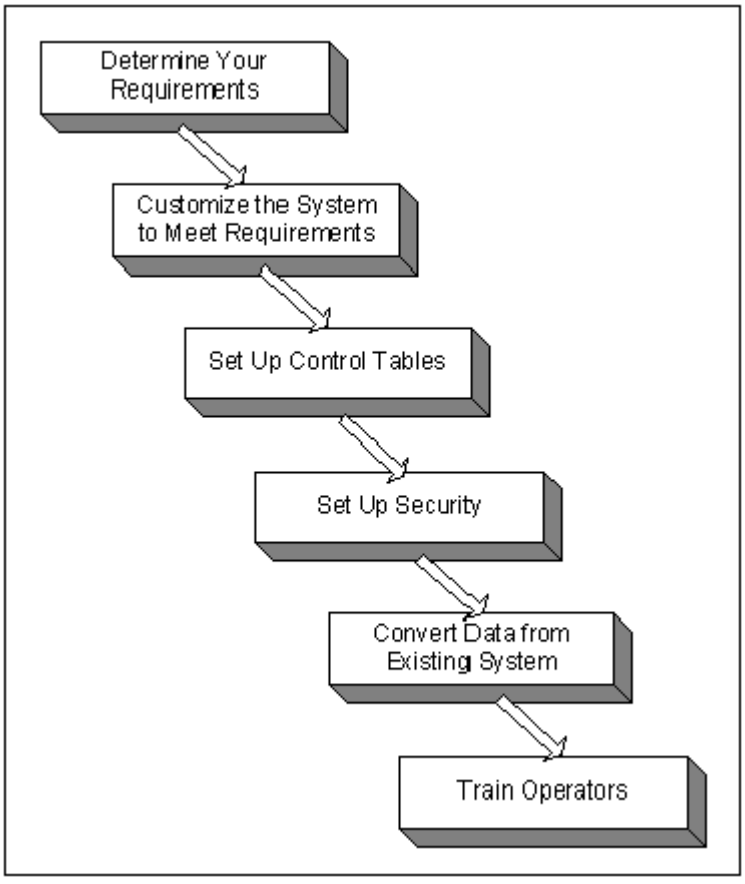
Chapter 1

Preparing To Implement

Topics:

Getting ready for production takes a good deal of planning. You have probably already begun analyzing your requirements according to your business and organizational needs. You will need to review your current environment and think about what changes could be made now and in the future. And while you might have decided to simply transfer your current processing structure to Oracle Revenue Management and Billing, you may also have discovered that Oracle Revenue Management and Billing can provide new options.

Because the system is sophisticated and customizable, there are a number of steps involved in rolling out and using your new system.



The topics in this section describe the order in which the control tables should be set up.

Control Table Setup Sequence

To implement the system, you must set up your organization's business rules in "control tables". Setting up these tables is time-consuming because we allow you to tailor many aspects of the system to meet your organization's requirements. We strongly recommend that you take the time to document how you plan to set up all of these tables before you use the following roadmap to enter the control data. Time spent understanding the interrelationships between this data will reap the rewards of a clean system that meets your current and long term needs.

While we describe the transactions and options in more detail in other sections of this manual, use the following chart (and the remaining sections of this chapter) as your roadmap. Here we list the order in which you perform tasks and the pages you'll use to set up your system. The order is important because some information must exist before other information can be defined (i.e., many dependencies exist).

Note: Auto setup. The Auto Setup column in the following table contains suggestions to save you time. It also indicates if a control table contains information when the system is installed.

Note: You don't have to set up every control table. You need only set up those control tables that govern functions that are applicable to your organization.

Function	Menu	Auto Setup
Global Context		
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that populates global context values. The global context is used by various zones in the system to display relevant data. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms**.	
Accounting Environment		
Country & State	Admin Menu, Country	
Currency	Admin Menu, Currency	USD is automatically populated
Accounting Calendar	Admin Menu, Accounting Calendar	
GL Division	Admin Menu, General Ledger Division	
Security Environment		
Application Service	Admin Menu, Application Service	All base package transactions are automatically populated
Security Type	Admin Menu, Security Type	

Function	Menu	Auto Setup
User Group	Admin Menu, User Group Note, you won't be able to set up users at this point	One user group, `ALL-SERVICES`, is automatically setup. It references all other application services and a single user called `SYSUSER`. Note, you may be able to *import user groups if your organization has already defined them using LDAP*.
Language	Admin Menu, Language	`ENG` is automatically populated
Display Profile	Admin Menu, Display Profile	Two display profiles are automatically setup: `NORTHAM` displays currencies and dates in a classic American format; `EURO` displays information in a classic European format
Data Access Role	Admin Menu, Data Access Role	
Access Group	Admin Menu, Access Group	
User	Admin Menu, User	`SYSUSER` is automatically set up. Note, you may be able to *import user groups if your organization has already defined them using LDAP*.
Return to User Group	You must return to your user groups and define all of their users	
Customer Class Environment		
Customer Class	Admin Menu, Customer Class. At this point, you'll only be able to set up your customer class codes. You will return to these customer classes throughout the setup process to populate additional information.	
Financial Transaction Environment		
Work Calendar	Admin Menu, Work Calendar	

Function	Menu	Auto Setup
Division	Admin Menu, Division	
Revenue Class	Admin Menu Revenue Class	
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithm that constructs a distribution code's corresponding GL account when it is interfaced to the general ledger	
Distribution Code	Admin Menu, Distribution Code	
Bank & Bank Accounts	Admin Menu, Bank	
Billable Charge Template	Admin Menu, Billable Charge Template. Note, if you want the system to default service quantities onto billable charges created using this template, you must setup the appropriate unit of measure code, time-of-use code and/or service quantity identifier.	
Billable Charge Line Type	Admin Menu, Billable Charge Line Type	
Algorithm	Admin Menu, Algorithm. You will need to set up several algorithms. These algorithms: 1) retrieve a bill segment's consumption, 2) calculate a bill segment's bill lines, 3) construct a bill segment's financial transaction, 4) cancel previously estimated bill segments	
Bill Segment Type	Admin Menu, Bill Segment Type	
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithm that constructs a payment segment's financial transaction	
Payment Segment Type	Admin Menu, Payment Segment Type	
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithm that constructs an adjustment's financial transaction	
Algorithm	Admin Menu, Algorithm. Several plug-in spots are available to perform additional logic when processing adjustments. For example, if you have the system calculate adjustments, you must set up an adjustment generation algorithm. Refer to *Adjustment Type* for other available plug-in spots that may be used by your implementation.	

Function	Menu	Auto Setup
Algorithm	Admin Menu, Algorithm. You may want to set up an algorithm that formats the Adjustment information that is displayed throughout the system for a specific Adjustment Type. This algorithm is plugged-in on the *Adjustment Type*.	
Algorithm	Admin Menu, Algorithm. You may want to set up an algorithm that formats the Adjustment information that is displayed throughout the system. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Adjustment Type	Admin Menu, Adjustment Type	
Adjustment Type Profile	Admin Menu, Adjustment Type Profile	
Approval Profile	Admin Menu, Approval Profile. Note, an approval profile references a To Do type and one or more To Do Roles; these must be set up before you can set up the approval profile. After the approval profile(s) are set up, they must be referenced on the adjustment types that they govern.	
Cancel Reason - Bill	Admin Menu, Bill Cancel Reason	
Cancel Reason - Payment	Admin Menu, Payment Cancel Reason	
Cancel Reason - Adjustment	Admin Menu, Adjustment Cancel Reason	
Tender Type	Admin Menu, Tender Type	
Tender Source	Admin Menu, Tender Source	
A/P Request Type	Admin Menu, A/P Request Type	

Function	Menu	Auto Setup
Installation	Admin Menu, Installation Options - Framework and Admin Menu, Installation Options. Many fields on the installation record impact the financial transaction environment. Refer to the description of the *Billing* and *Financial Transaction* tabs for more information. Also, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Messages**.	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that distributes payments.	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that handles overpayment situations.	
Algorithm	Admin Menu, Algorithm. You may need to set up an algorithm if specific customers can have individual bill due dates.	
Algorithm	Admin Menu, Algorithm. You may need to set up an algorithm if you want the system to delete bills that contain only information about historical payments.	
Algorithm	Admin Menu, Algorithm. You may need to set up an algorithm if you want the system to levy a non-sufficient funds charge if a payment is canceled due to non-sufficient funds.	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that formats the bill information that is displayed throughout the system. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	

Function	Menu	Auto Setup
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that formats the payment information that is displayed throughout the system. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that defaults the amount when a payment is manually added. This algorithm also calculates the amount of an automatic payment for a bill for an account with an active auto pay option. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms**.	
Algorithm	Admin Menu, Algorithm. Refer to *Customer Class* for other available plug-in spots that may be used by your implementation to perform additional logic when processing payments and bills.	
Return to Customer Class	Admin Menu, Customer Class. You will need to plug-in the algorithms defined above on your customer classes.	
Budget Environment		
Algorithm	Admin Menu, Algorithm. You will need to set up several algorithms at this time: How To Calculated The Recommended Budget Amount, How To Periodically True Up A Customer's Budget Amount, The Circumstances When The System Should Highlight A Customer As Having An Anomalous Budget.	
Budget Plan	Admin Menu, Budget Plan	

Function	Menu	Auto Setup
Algorithm	Admin Menu, Algorithm. Budget eligibility is set at the contract type level. You will need to set up an override budget eligibility algorithm if some contracts for a contract type are not eligible for budget based on certain conditions.	
Customer Environment		
Account Management Group	Admin Menu, Account Management Group. Note, you will probably have to set up To Do Type and To Do Roles before you can setup account management groups. Refer to *Assigning A To Do Role* for more information on how account management groups may be used to define an entry's role.	
Account Relationship Type	Admin Menu, Account Relationship Type	
Alert Type	Admin Menu, Alert Type	
Bill Message	Admin Menu, Bill Message	
Algorithm	Admin Menu, Algorithm. If you have software that's capable of reconstructing an image of a bill in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your bill image software.	
Bill Route Type	Admin Menu, Bill Route Type	
Contract Quantity Type	Admin Menu, Contract Quantity Type	
Algorithm	Admin Menu, Algorithm. If you have software that's capable of reconstructing an image of a letter in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your letter image software.	
Letter Template	Admin Menu, Letter Template	
Customer Contact Class	Admin Menu, Customer Contact Class	
Customer Contact Type	Admin Menu, Customer Contact Type	
Algorithm	Admin Menu, Algorithm. You may need to set up the algorithms that determine if person ID's are in a predefined format.	

Function	Menu	Auto Setup
Person Identifier Type	Admin Menu, Person Identifier Type	
SICs	Admin Menu, SIC Code	
Tax Exempt Type	Admin Menu, Tax Exempt Type	
Algorithm	Admin Menu, Algorithm. You may need to set up the algorithms that determine if phone numbers are in a predefined format.	
Phone Type	Admin Menu, Phone Type.	
Person Relationship Type	Admin Menu, Person Relationship Type.	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that formats the person information that is displayed throughout the system. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm to validate a person's name. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms**.	
Algorithm	Admin Menu, Algorithm. You can override the system's standard account information string by setting up an algorithm that produces this string of information. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms**.	

Function	Menu	Auto Setup
Algorithm	Admin Menu, Algorithm. If you have software that's capable of reconstructing an image of a letter in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Algorithm	Admin Menu, Algorithm. If you have software that's capable of reconstructing an image of a bill in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Installation	Admin Menu, Installation Options. Many fields on the installation record impact the Customer Environment. Refer to the description of the *Main*, *Person*, and *Account* tabs for more information.	
Statements		
Algorithm	Admin Menu, Algorithm. If you have software that's capable of reconstructing an image of a statement in a PDF (for the purpose of online display), you will need to create an algorithm that formats the extract records that are sent to your statement image software.	
Statement Route Type	Admin Menu, Statement Route Type	
Statement Cycle	Admin Menu, Statement Cycle	

Function	**Menu**	**Auto Setup**
Algorithm	Admin Menu, Algorithm. If you have software that's capable of reconstructing an image of a statement in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Automatic Payment (EFT) Environment		
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm to create automatic payments. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms**.	
Tender Source	Admin Menu, Tender Source Note: earlier, you created tender sources for the remittance processor and your cash drawers. At this point, you'll need to add at least one tender source for automatic payments. Why? Because automatic payments get linked to a tender control (which, in turn, gets linked to a tender source) when they are interfaced out of the system.	
Algorithm	Admin Menu, Algorithm. You will need to set up the appropriate automatic payment date calculation algorithm to populate the extract, GL interface and payment dates on automatic payments.	
Auto Pay Route Type	Admin Menu, Auto Pay Route Type	

Function	Menu	Auto Setup
Tender Type	Admin Menu, Tender Type Note: earlier, you created tender types for things like cash, checks, etc. At this point, you'll need to add a tender type for each type of automatic payments (e.g., direct debt, credit card, etc.).	
Work Calendar	Admin Menu, Work Calendar. You need only set up additional work calendars if the auto pay sources (i.e., the financial institutions) have different working days than does your organization	
Algorithm	Admin Menu, Algorithm. If you need to validate the customer's bank account or credit card number, you will need to set up the appropriate validation algorithms.	
Auto Pay Source Type	Admin Menu, Auto Pay Source Type	
Algorithm	Admin Menu, Algorithm. You may need to set up an algorithm if your customers can define a maximum withdrawal limit on their autopay options.	
Return to Customer Class	Admin Menu, Customer Class. You should plug-in the Autopay Over Limit Algorithm in each appropriate customer class.	
Deposit Environment		
Algorithm	Admin Menu, Algorithm. You will need to set up several algorithms at this time: The Definition Of A Good Customer, When To Refund A Deposit To A Customer, When To Recommend An Additional Deposit, How / When To Calculate Interest, How To Generate The Recommended Deposit Amount.	
Deposit Class	Admin Menu, Deposit Class	
Non Cash Deposit Type	Admin Menu, Non Cash Deposit Type	
Credit & Collections Environment (if you collect on overdue bills (as opposed to overdue debt), you will NOT set up these tables; refer to *Overdue Processing - Set Up Tasks* for the list of control tables required to collect on overdue bills)		
Algorithm	Admin Menu, Algorithm. You may need to set up algorithms if you have non-standard collection events.	
Collection Event Type	Admin Menu, Collection Event Type	

Function	Menu	Auto Setup
Algorithm	Admin Menu, Algorithm. You may need to set up a collection process cancellation algorithm if your organization allows individual contracts to be removed from a collection process if they are paid (rather than performing cancellation based on all contracts in a debt class).	
Collection Process Template	Admin Menu, Collection Process Template	
Collection Class	Admin Menu, Collection Class	
Algorithm	Admin Menu, Algorithm. You will need to set up several algorithms at this time: Collection process cancellation criteria and Override arrears due to promise to pay.	
Debt Class	Admin Menu, Debt Class	
Write Off Debt Class	Admin Menu, Write Off Debt Class	
Algorithm	Admin Menu, Algorithm. You will need to set up Collection Condition algorithms.	
Collection Class Control	Admin Menu, Collection Class Control	
Algorithm	Admin Menu, Algorithm. You will need to set up several algorithms at this time: How to refer debt to a collection agency, How to transfer debt to another active contract, How to write down small amounts of debt, and How to refund credit balances to a customer.	
Algorithm	Admin Menu, Algorithm. You may need to set up algorithms if you have non-standard write-off events.	
Write Off Event Type	Admin Menu, Write Off Event Type (Note, you'll have to wait until you have defined your Contract Types before you can set up the Write Off Events because Contract Type is a necessary parameter to write off debt).	
Write Off Process Template	Admin Menu, Write Off Process Template	
Write Off Control	Admin Menu, Write Off Control	

Function	Menu	Auto Setup
Collection Agency	Admin Menu, Collection Agency. Note, each collection agency references a person therefore you must set up a person for each agency before you can enter collection agency information.	
Algorithm	Admin Menu, Algorithm. You may need to set up algorithms if you have special logic that should be executed when a promise to pay is canceled.	
Promise To Pay Type	Admin Menu, Promise To Pay Type	
Pay Method	Admin Menu, Pay Method	
Third Party Payor	Admin Menu, Third Party Payor. Note, you must create an account before you can create a third party payor.	
Installation	Admin Menu, Installation. Several fields on the *installation record* impact the Credit & Collections Environment.	
Algorithm	Admin Menu, Algorithm. You will need to setup an algorithm that's called when a user write-off debt real time.	
Return to Customer Class	Admin Menu, Customer Class. You should plug-in the Autopay Over Limit Algorithm in each appropriate customer class.	
Services & Characteristics		
Service Type	Admin Menu, Service Type	
Algorithm	Admin Menu, Algorithm. If you have ad hoc characteristic types, you may need to set up the algorithms that control how they are validated	
Foreign Key Reference	Admin Menu, FK Reference. If you have foreign key characteristic types, you may need to set up foreign key references to control how the user selects the characteristic values (and how the foreign key values are validated).	All base package FK references are automatically populated
Characteristic Type & Values	Admin Menu, Characteristic Type	
Bill Environment		
Bill Cycle	Admin Menu, Bill Cycle	

Function	**Menu**	**Auto Setup**
Bill Period	Admin Menu, Bill Period	
Bill Route Type	Admin Menu, Bill Route Type	
Rate Environment		
Frequency	Admin Menu, Frequency	
Service Quantity Identifier	Admin Menu, Service Quantity Identifier	
Algorithm Type	Admin Menu, Algorithm Type. If you create new Service Quantity Rules you must set up an algorithm type for each such rule (the algorithm type defines the types of parameters that are passed to the SQ rule).	All base package algorithm types are automatically populated
Service Quantity Rule	Admin Menu, Service Quantity Rule	
Bill Factor	Main Menu, Rates, Bill Factor	
Algorithm Type	Admin Menu, Algorithm Type. If you create new Register Rules you must set up an algorithm type for each such rule (the algorithm type defines the types of parameters that are passed to the register rule).	All base package algorithm types are automatically populated
Rate	Main Menu, Rates, Rate Schedule	
Rate Version	Main Menu, Rates, Rate Version	
Algorithm	Admin Menu, Algorithm. If you use algorithms to dynamically change step boundaries, calculate prices, or implement rate component eligibility rules, you must set up these algorithms.	
Rate Component	Main Menu, Rates, Rate Component	
Bill Factor Value	Main Menu, Rates, Bill Factor Value	
Bill Factor Interval Values	Main Menu, Rates, BF Interval Values	
Late Payment Environment		
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithm that determine if customers in a customer class are eligible for late payment charges	
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithm that levies late payment charges for customers in a customer class	

Function	Menu	Auto Setup
Return to Customer Class	Admin Menu, Customer Class. You will need to plug-in the late payment charge algorithms set up above.	
Contract Configuration		
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithms that determine: How to calculate the late payment charge amount for contracts of a given type Special criteria to be tested before a contract is severed. Special processing that should take place prior to the completion of a bill that references contracts of a given type. Special processing that should take place during completion of a bill that references contracts of a given type. Special processing that should take place when contracts of a given type are created. Special processing that should take place when a financial transaction is frozen for contracts of a given type.	
Algorithm	Admin Menu, Algorithm. You may want to set up an algorithm that formats the contract information that is displayed throughout the system. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Algorithm	Admin Menu, Algorithm. You may want to set up an algorithm that formats the contract information that is displayed throughout the system for a specific Contract Type. This algorithm is plugged-in on the *Contract Type - Algorithms*.	

Function	**Menu**	**Auto Setup**
Algorithm	Admin Menu, Algorithm. If you want a Control Central alert to highlight when the current account has any `stopped` contract(s), you will need to set up the algorithm that does this. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Contract Type	Admin Menu, Contract Type	
Terms and Conditions	Admin Menu, Terms and Conditions	
Notification and Workflow		
Workflow Event Type	Admin Menu, Workflow Event Type	
Workflow Process Template	Admin Menu, Workflow Process Template	
Notification Upload Type	Admin Menu, Notification Upload Type	
Workflow Process Profile	Admin Menu, Workflow Process Profile	
Notification External (Sender) ID's	Admin Menu, Notification External ID	
Notification Download Type	Admin Menu, Notification Download Type	
Service Provider.	Admin Menu, Service Provider. Note, you must create a person before you can create a service provider.	
Notification Download Profile	Admin Menu, Notification Download Profile	

Function	Menu	Auto Setup
Algorithm	Admin Menu, Algorithm. If you want a Control Central alert to highlight when the current account has `active` workflow processes, you will need to set up the algorithm that does this. This algorithm is plugged-in on the installation options. For more information, refer to **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Defining General Options** > **Defining Installation Options** > **Installation Options - Algorithms** .	
Sales and Marketing		
Order Hold Reason	Admin Menu, Order Hold Reason	
Order Cancel Reason	Admin Menu, Order Cancel Reason	
And more...	Refer to *Campaign and Package Setup Sequence* for additional setup requirements	
Service Credit Membership		
Algorithm	Admin Menu, Algorithm. You may need to set up algorithms for the service credit membership type and service credit event type to control behavior for the service credit membership and its events.	
Credit Unit	Admin Menu, Credit Unit. If your service credits record non-monetary units.	
Service Credit Membership Type	Admin Menu, Service Credit Membership Type	
Service Credit Event Type	Admin Menu, Service Credit Event Type	
Membership Inactive Reasons	Admin Menu, SC Membership Inactive Reason	
Wrap Up		
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithms that determine: Special alerts on Control Central (assuming you have special alerts)	

Function	Menu	Auto Setup
Installation Options	Admin Menu, Installation Options - Framework and Admin Menu, Installation Options. At this point, it's a good idea to double-check everything on the installation record.	
Postal Default	Admin Menu, Postal Code Default	

If you have cash drawers you will also need to set up the following information:

- Create a person / account to which you will link your over / under contract. Refer *How To Get An Unbalanced Tender Control In Balance (Fixing Over/Under)* for more information.
- Create a contract to which your over/under payments will be linked. This contract will reference your over / under contract type. Refer to *Over / UnderCashDrawer Segmentation* for more information.

If you upload payments from an external source (e.g., a remittance processor or lock box), you must set up the following information:

- Create a person and account to which the system will link payments with invalid account. Refer to *Phase 3 - CreatePayment Events, Tenders, Payments and Payment Segments* for information about the process that books invalid payments to this account. Refer to *How To Transfer A Payment From One Account To Another* for how a user transfers the payment from the invalid account to the correct account (once known).
- Create a contract for this account. This contract will reference your payment suspense contract type. The system needs this contract so that it can distribute the invalid account's payment (and this is necessary so that cash will reflect the payment). Refer to *Payment Upload Error Segmentation* for more information.
- Update the tender source associated with the respective source of payments to indicate the contract created in the previous step should be used for payments with invalid accounts. Refer to *SettingUp Tender Sources* for more information.
- Because the payment upload process simply books payments that reference invalid accounts to the account associated with the suspense contract on the payment's tender source, this account should belong to a customer class with the appropriate payment distribution algorithms. This may entail creating a new customer class that will only be used on suspense accounts. This customer class would need the following algorithms:
  - We'd recommend using a simple payment distribution algorithm like *PYDIST-PPRTY* (distribute payment based on contract type's payment priority)

    .
  - We'd recommend using an overpayment distribution algorithm like *OVRPY-PPRTY* (distribute overpayment to highest priority contract type)

    .

The remaining sections describe additional control tables that must be set up for specific functional areas.

Cross Reference To The Remaining Chapters

The table in the previous section describes the order in which you should enter your control tables. These tables are described at length in the following chapters.

- Refer to *Defining General Options Addendum* and **Defining General Options** (under**Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide**) for a discussion of the control tables associated with general functionality (e.g., country codes, state codes, etc.).
- Refer to Defining Financial Transaction Options for a discussion of the tables affecting your financial transactions (e.g., bill segment types, payment segment types, etc.)

- Refer to *Defining Customer Options* for a discussion of the control tables affecting persons, accounts and contracts.
- Refer to Defining Credit & Collections Options for a discussion of the control tables affecting your collection activities.
- Refer to *Rates* for a discussion of the control tables affecting your rates.
- Refer to *Defining Contract Type Options* for a discussion of the control tables affecting your contract types.
- Refer to the **Background Processes** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide**) for a discussion of the control tables affecting your background processes.
- Refer to the **Defining Algorithms** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide**) for a discussion of the control tables affecting the algorithms referenced on many control tables.
- Refer to *Workflow and Notification Options* for a discussion of the control tables affecting the processing of notifications to and from service providers.
- Refer to *Statements* for a discussion of the tables affecting the statement setup options for your customers.
- Refer to *Defining Service Credit Options* for a discussion of the tables affecting the service credit membership setup options for your customers.

Open-Item Accounting Table Setup Sequence

Open-item accounting tables need only be set up if your organization practices *Open Item Accounting*. Refer to *Setting Up The System To Enable Open Item Accounting* for a description of the tables that must be set up to enable this functionality.

Fund Accounting Table Setup Sequence

Fund accounting tables need only be set up if your organization practices *Fund Accounting*. Refer to *Setting Up The System To Enable Fund Accounting* for a description of the tables that must be set up to enable this functionality.

Payment Event Distribution Table Setup Sequence

Payment event distribution tables need only be set up if your organization opted to use the distribution rules method to create payment events. Refer to *Setting Up The System To Use Distribution Rules* for a description of the tables that must be set up to enable this functionality.

Loans Table Setup Sequence

Loans need only be set up if your organization offers *loans* to your customers. Refer to *Setting Up The System To Enable Loans* for a description of the tables that must be set up to enable this functionality.

Quotes Table Setup Sequence

Quotes need only be set up if your organization sends quotes to customers or prospects. Refer to *Defining Quotation Options* for a description of the tables that must be set up to enable this functionality.

Scripting Table Setup Sequence

Scripts need only be set up if your organization opts to create scripts to step your users through business processes. Refer to *Defining Script Options* for information about scripting and the tables that must be set up to enable this functionality.

Reports Setup Sequence

In order to use the reporting tool, you will need to set up reporting options. Refer to the **Configuring The System To Enable Reports** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Reporting and Monitoring Tools** > **Reporting Tool Integration**) for more information.

XML Application Integration Setup Sequence

In order to use the XAI tool for sending information between third parties, you will need to set up XAI control tables. Refer to the **XML Application Integration** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **External Messages**) for more information.

Case Management Setup Sequence

Case management options need only be set up if your organization uses cases to manage issues. Refer to *Setting Up Case Types* for more information.

Batch Scheduler Setup Sequence

Batch scheduler options need only be set up if your organization uses the batch scheduling functionality provided by the system rather than a third party batch scheduling system. Refer to *Setting Up The Batch Scheduler* for more information.

Zone Set Up

Most zones are delivered with the base-package and do not require any configuration. However, some zones are only available if configured by your implementation. Refer to *Configuring Zones* for more information.

To Do Options Setup

Refer to the **Setting Up To Do Options** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **To Do Lists**) for more information on how to configure the system to match your organization's To Do management needs.

Chapter 2

Defining General Options Addendum

Topics:

This section describes control tables that are used throughout Oracle Revenue Management and Billing.

Defining Installation Options

The topics in this section describe the various installation options that control various aspects of the system that are specific to the Oracle Revenue Management and Billing product.

Fastpath: Refer to *Installation Options - Framework* for options that are common to products on the same framework.

Installation Options - Main

Select **Admin Menu, Installation Options** and use the **Main** tab to define system-wide installation options.

Description of Page

Use **Quick Add Tender Type** to define the tender type defaulted on payments added using the *Payment Quick Add* transaction.

Use **Starting Balance Tender Type** to define the tender type of the starting balance recorded on your tender controls (this will almost always be the tender type associated with "cash"). This value is used during tender control balancing as a separate balance is required for each tender type in order to balance a tender control. Refer to *The Lifecycle Of A Tender Control* for more information.

Fastpath: For more information, refer to *Setting Up Tender Types*.

If you use orders to create new customers, define the **Campaign** that should be defaulted on orders created when the order transaction is opened for a new customer. Refer to *Real time Marketing of Services to a New Customer* for more information.

The **Alternate Representation** flag should be set to `None` unless your organization uses multiple character sets for a person's main name. Alternate representations are typically only used in countries where multiple character sets are used. For example,

- In Hong Kong, a person's name may be written in both Chinese characters and in English.
- In Japan, a person's name may be written in both Kanji and Katakana.

In both of the above situations, users need to be able to use both representations to find a customer.

If you want to use alternate representation for a person's primary name, set this flag to Name.

The following points describe the ramifications of this flag in the system:

- If you support alternate representations of a person's primary name,
  - The name grid on *Person - Main* allows you to specify an `Alternate` name for the person.
  - If you use the base package *name formatting algorithm*, a person's name will be shown throughout most of the system in the format AAA (BBB), where AAA is the person's primary name and BBB is the person's alternate name. Note, this format does not apply to names that appear in search results (i.e., the alternate name is not concatenated to the main name in search results; however you can search for information using the alternate name).
  - Most of the system's person name-oriented searches will allow users to use both a person's primary and alternate names to search for information.

Set the **CTI Integration** flag to `Yes` if your organization integrates with an external computer telephony integration (CTI) system that supports a "get next caller in the queue" function. If this flag is set to `Yes`, then *Next Call* button will appear in the action toolbar allowing customer service representatives to request the next customer waiting in the queue to speak to a CSR.

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 this field's option and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. Refer to *Caching Overview* **for information** on how to clear the system login cache (this is the cache in which installation options are stored).

Installation Options - Person

Select **Admin Menu, Installation Options** and use the **Person** tab to define person-specific installation options.

Description of Page

Use the **Person ID Usage** to indicate whether or not at least one form of identification is `Required` or `Optional` when a new person is added.

Each form of identification has an identifier type. For persons that are humans (as defined by the person type), the system defaults the identifier type defined in **Identifier Type (Person)**. For persons that are businesses (as defined by the person type), the system defaults the identifier type defined in **Identifier Type (Business)**.

Installation Options - Account

Select **Admin Menu, Installation Options** and use the **Account** tab to define account-specific installation options.

Description of Page

When a new account is added, the system requires it have a customer class. If the main customer linked to the account is a human (as defined by the customer's person type), the system defaults the customer class defined in **Customer Class (Person)**. For persons that are businesses (as defined by the person type), the system defaults the customer class defined in **Customer Class (Business)**. For more information, refer to *Setting Up Customer Classes*.

In addition to requiring a customer class when a new customer is added, the system also requires a "main customer" (i.e., a reference to a person who is identified as the main customer for the account). Enter the default **Account Relationship Type Code** to be used to define the main customer relationship. For more information, refer to *Setting Up Account Relationship Codes*.

Enter the default **Bill Route Type** to be used to define how bills should be routed to a customer. For more information, refer to *Setting Up Bill Route Types*.

Enter the default **Quote Route Type** to be used to define how quotes should be routed to a customer. For more information, refer to *Setting Up Quote Route Types*.

If the number of pending start and pending stop contracts exceeds the **Start Stop Detail Threshold** for an account, it is considered a large account for start stop purposes. Refer to *Start/Stop Maintenance* for more information.

Installation Options - Billing

Select **Admin Menu, Installation Options** and use the **Billing** tab to define billing-specific installation options.

Description of Page

The **Bill Segment Freeze Option** controls when a contract's balance and the general ledger is affected by bill segments and certain types of adjustments. Refer to *Preventing Contract Balances And The GL From Being Impacted Until Bill Completion* to understand the significance of this option.

The **Accounting Date Freeze Option** controls how the accounting date defined on financial transactions is populated. Refer to *Forcing The Freeze Date To Be Used As The Accounting Date* to understand the significance of this option.

Define the **Minimum Amount for Final Bill**. If a final bill is less than this amount, the bill is still produced; it's just not printed.

Typically, the system sets a bill's Bill Date equal to the date on which it is completed. If you want to be able to specify a bill's Bill Date when you complete a bill, turn on **User Can Override Bill Date**. You would only want to override the bill date if you are setting up sample bills from historical period whose bill date needs to reflect the respective historical period.

Select **Use Sequential Bill Numbers** if you require a system-assigned unique *sequential* number for each bill in addition to the Bill Id, which is a system-assigned *random* number used as the bill's primary identifier. Refer to the *Alternate Sequential Bill Numbers*section for more information.

The **Bill Correction Option** lets you control whether your implementation uses Credit Notes or Correction Notes. Select the **Credit Note** option if you require bill segment cancellation details to be presented to the customer on a separate bill (referred to as a credit note). Refer to *Credit Notes* for more information. Select the **Correction Note** option if you require bill segment cancellation details and bill segment rebill details to be presented to the customer on a separate bill (referred to as a correction note). Refer to *Correction Notes* for more information.

Note: Credit Notes or Correction Notes. The Bill Correction option on the Installation table controls whether Credit Notes or Correction Notes are allowed. If your implementation uses Correction Notes, the override label on the following should be customized accordingly:

Note: Lookup value `CRNT` on the customizable lookup field `TXN_FLTR_TYPE_FLG` (this lookup value is used on the Match Event Page and Account Bill History transactions)

Note: Lookup value `CR` on the customizable lookup field `PYCAN_SYS_DFLT_FLG` (this lookup value is used on the Pay Cancel Reason transaction)

Note: Metadata field `CR_NOTE_FR_BILL_ID` (this field is used on the Bill Search Page)

The **Autopay Creation Option** controls when automatic payments are created, distributed, and frozen. This option allows you to control when automatic payments will affect customer's balances and when their financial impact affects the general ledger. Refer to *How And When Are Automatic Payments Created* to understand the significance of this option.

The **Sequential Invoice** field indicates whether you want to generate alternate sequential bill numbers unique throughout the system or within the division. The valid values are:

- **Division-specific** — Used when you want to generate alternate sequential bill numbers unique within the division.
- **System-wide** — Used when you want to generate alternate sequential bill numbers unique throughout the system.

Sequence Generation Algorithm — Used to attach an algorithm which indicates how to sequence alternate bill numbers which are unique throughout the system. Refer to the *Sequential Bill Number Generation Algorithms* section for more information.

Installation Options - C&C

Select **Admin Menu, Installation Options** and use the **C&C** tab to define credit and collections-specific installation options.

Description of Page

When you look at an account or contract's debt, the system shows the respective age of each piece of outstanding debt. The **Oldest Bucket Age (Days)** defines the debt age after which the system groups all outstanding debt together. For example, if this field is `180`:

- The exact age of each element of debt that is less than 180 days old would be shown as a separate line item in the aged debt information.
- All debt older than 180 days would be amalgamated into a single "bucket".

Oldest Bucket Age (Days) also has another use - it defines the age of financial transactions that are considered by the background process that marks old debt as "redundant". This batch process is referred to by the batch code of `REDSAAMT`. Please refer to *Process What's Ready Background Processes* for more information about this process.

Caution: If you change the value of **Oldest Bucket Age (Days)** after debt has been marked as "redundant" by `REDSAAMT`, the system will NOT re-age the old debt (i.e., once a financial transaction has been marked as "redundant", it is "redundant" forever).

Enter what you consider to be an excellent credit rating in **Beginning Credit Rating**. Collection events can cause an account's credit rating to decrease. When an account's credit rating falls below a certain level, different collection processes may ensue.

Use **Beginning Cash-Only Score** to define the cash-only score for accounts with a perfect payment history (i.e., one without non-sufficient funds). When you cancel a payment tender and use a cancellation reason marked as NSF, the system will cause the account's cash-only score to increase by the value on the payment cancellation reason.

Use **Credit Rating Threshold** to define when an account's credit rating becomes risky. When an account's credit rating falls beneath the Credit Rating Threshold, the system will:

- Assuming you've enabled the Control Central alert algorithm, *C1-CRRT-ACCT*, an alert displays when an account's credit rating falls below the credit rating threshold on the CIS installation table. This algorithm is plugged-in on the *installation record*.
- Subject the account's debt to different collection criteria. For more information, refer to *Designing Your Collection Class Control Overrides*.

Use **Cash-Only Threshold** to define the number of cash-only points a customer must have before the system warns the CSR accepting payments that the account is cash-only.

Installation Options - Financial Transaction

Select **Admin Menu, Installation Options** and use the **Financial Transaction** tab to define financial transaction installation options.

Description of Page

Use **G/L Batch Code** to define the batch process that is used to interface your financial transactions to your general ledger. The process is snapped on FT download records by the GLS background process.

Use **A/P Batch Code** to define the batch process that is used to interface your check requests (initiated with adjustments with an adjustment type that reference an A/P request type) to you accounts payable system.

Use **Fund Accounting** to indicate if *fund accounting* is `Practiced` or `Not Practiced` at your organization.

Use **Alternate Currency** to indicate if your organization accepts customer payments in currencies other than the account's currency. Refer to *Alternate Currency Payments*to understand the significance of this option.

Installation Options - Algorithms

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

System Event	Optional / Required	Description
`Account Information`	Optional	We use the term "Account information" to describe the basic information that appears throughout the system to describe an account. The data that appears in "account information" is constructed using this algorithm. Plug an algorithm into this spot to override the system default "Account information". Click *here* to see the algorithm types available for this system event.
`Adjustment Information`	Optional	We use the term "Adjustment information" to describe the basic information that appears throughout the system to describe an adjustment. The data that appears in "Adjustment information" is constructed using this algorithm. Plug an algorithm into this spot to override the system default "Adjustment information". Note: This algorithm may be further overridden by an "Adjustment information" plug-in on the Adjustment Type. Refer to *Adjustment Type* for how algorithms of this type are used. Click *here* to see the algorithm types available for this system event.
`Automatic Payment Creation`	Required if you allow customers to *pay automatically*	This algorithm is executed to create automatic payments whenever the system creates automatic payments. Refer to *How And When Are Automatic Payments Created* for the details. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Bill Information`	Required	We use the term "Bill information" to describe the basic information that appears throughout the system to describe a bill. The data that appears in "bill information" is constructed using this algorithm. Click *here* to see the algorithm types available for this system event.
`Case Information`	Optional	We use the term "Case information" to describe the basic information that appears throughout the system to describe a case. The data that appears in "case information" is constructed using this algorithm. Plug an algorithm into this spot to override the system default "Case information". Note: This algorithm may be further overridden by a "Case information" plug-in on the Case Type. Refer to *Case Type* for how algorithms of this type are used. Click *here* to see the algorithm types available for this system event.
`Collection Agency Referral Information`	Optional	We use the term "Collection Agency Referral information" to describe the basic information that appears throughout the system to describe a collection agency referral. Plug an algorithm into this spot to override the system default "collection agency referral information". Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Collection Process Additional Information`	Optional	This algorithm displays additional information related to a collection process in a special field on the *collection process* main page. Click *here* to see the algorithm types available for this system event.
`Control Central Alert`	Optional	There are two types of alerts that appear on *Payment Event - Main*: 1) hard-coded system alerts and 2) alerts constructed by plug-in algorithms. You cannot change the hard-coded alerts. However, by plugging in this type of algorithm you can introduce additional alerts. An error displays if more than 60 alerts are generated for an account by plug-in algorithms. Click *here* to see the algorithm types available for this system event.
`Credit Rating "Created By" Information`	Required	The data that appears in the credit rating "created by" information is constructed using this algorithm. Refer to *Account - C&C* for more information about the credit rating. Click *here* to see the algorithm types available for this system event.
`Credit Rating History Information`	Optional	We use the term Credit Rating History information to describe the basic information that appears throughout the system to describe a *credit rating history* entry. Plug an algorithm into this spot to override the system default "credit rating history information". Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Determine Open Item Bill Amounts`	Required if you use overdue functionality to *collect on bills*	This algorithm is responsible for determining the unpaid amount of an open-item bill. It can also be used to return the unpaid amount for a specific contract on a bill. Click *here* to see the algorithm types available for this system event.
`Document Number`	Optional	If document numbers have been enabled on the installation record, this algorithm type assigns a document number to a bill or payment event. Click *here* to see the algorithm types available for this system event.
`Document Number Details`	Optional	If document numbers have been enabled on the installation record, this algorithm type is responsible for returning the details used to construct the document number. Click *here* to see the algorithm types available for this system event.
`Online Bill Display`	Optional	This algorithm constructs a PDF that contains the image of a bill. This algorithm is executed when the Display Bill button is clicked on the *Bill* page. Refer to *Technical Implementation of Online Bill Image* for more information. Click *here* to see the algorithm types available for this system event.
`Online Letter Image`	Optional	This algorithm constructs a PDF that contains the image of a letter. This algorithm is executed when the Display Letter button is pressed on *Customer Contact - Main*. Refer to *Technical Implementation of Online Letter Image* for more information. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Online Quote Image`	Optional	This algorithm constructs a PDF that contains the image of a quote. This algorithm is executed when the Display Quote button is pressed on *Quote - Main*. Refer to *Technical Implementation of Online Quote Image* for more information. Click *here* to see the algorithm types available for this system event.
`Online Statement Image`	Optional	This algorithm constructs a PDF that contains the image of a statement. This algorithm is executed when the Display Statement button is pressed on *Statement - Main*. Refer to *Technical Implementation of Online Statement Image* for more information. Click *here* to see the algorithm types available for this system event.
`Payment Amount Calculation`	Required	This algorithm is executed to calculate the amount of an automatic payment for a bill for an account with an active auto pay option. Refer to *How And When Are Automatic Payments Created* for more information on automatic payments. This algorithm is also executed to default the amount of a manually added payment. Refer to *How To Add A New Payment Event* for more information on adding a payment manually. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Payment Information`	Required	We use the term "payment information" to describe the basic information that appears throughout the system to describe a payment. The data that appears in "payment information" is constructed using this algorithm. Click *here* to see the algorithm types available for this system event.
`Person Information`	Required	In most parts of the system, a person's `Main` name is displayed to describe a person. However, several transactions do not use this method. Rather, these transactions call the algorithm that's plugged into this spot to construct the person's name. Refer to the description of the **Alternate Representation** flag on the **Main** tab for a list of these transactions and for the rationale behind this algorithm. Click *here* to see the algorithm types available for this system event.
`Person Name Validation`	Required	The format of names entered on *Person - Main* and *Order - Main* is validated using this algorithm. Click *here* to see the algorithm types available for this system event.
`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* on-line. This algorithm is used by the two on-line reporting pages to properly invoke the reporting tool from within the system. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Contract Information`	Optional	We use the term "Contract information" to describe the basic information that appears throughout the system to describe a contract. The data that appears in "Contract information" is constructed using this algorithm. Plug an algorithm into this spot to override the system default "Contract information". Note: This algorithm may be further overridden by an "Contract information" plug-in on the Contract Type. Refer to *Contract Type - Algorithms* for how algorithms of this type are used. Click *here* to see the algorithm types available for this system event.

Defining Customer Languages

As described under *Defining Languages*, you define the language in which each user see the system. In addition to defining each user's language, the system allows you to define each customer's preferred language. For example, one customer can receive bills in English whereas another customer could receive their bills in Chinese.

Each customer's language is defined by the *language code* on their *person record*. Bills, adjustments and other system-generated records will then be done in the language of the main customer of the account. In addition, the language code is passed on to all customer-facing interfaces, such as letter requests and bill print.

Note: You can define Rates in multiple languages - when a bill is generated, the line-item descriptions are generated and stored in the account's main customer's language of choice. Any one who subsequently views these bills can only see the descriptions in that language.

Note: To support bills and other correspondence, you must also provide translations of standard bill stock and letters. This must be handled by your printing software vendor.

Defining Accounting Calendar

Accounting calendar determine the accounting period to which a financial transaction will be booked. The following points describe how the system determines a financial transaction's account period:

- Every financial transaction references an accounting date and its contract
- Every contract references a contract type
- Every contract type references a GL division
- Every GL division references an accounting calendar
- The accounting calendar contains the cross reference between the accounting date specified on the financial transaction and related accounting period in your general ledger

Caution: This information must be the same as the information in your financial database.

To add or review an accounting calendar, choose **Admin Menu, Accounting Calendar**.

Description of Page

Enter a unique **Calendar ID** and **Description** for the calendar.

Enter the **Number Of Periods** for the calendar. Don't count the adjustment period, if you use one, or any special "system" periods.

Specify the **Fiscal Year**, each **Accounting Period** in that year, a **Period Description**, the **Begin Date** and the **End Date**.

When you enter begin and end dates, you can define monthly calendar periods or any fiscal period that matches your accounting calendar (weekly, bimonthly) as long as the begin and end dates of successive periods do not overlap. Every day of the year must be included in a period; do not leave gaps between period dates.

For each fiscal period, enter the **Open From Date** and **Open To Date**. These dates define when that particular business dates are open for posting financial transactions to that fiscal period. For example, you might calculate a bill on Sept 1 for usage recorded on 31 August. To post this financial transaction in the August period, you must keep it open through Sept 1.

As time passes, you will need to return to this transaction to manually enter ensuing years. You can enter several years at a time or incorporate the task into end-of-year system maintenance.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CAL_GL*

Defining General Ledger Divisions

There are two types of divisions referenced in the system: a GL Division and a Division. This is a rather powerful structure, but it can be confusing.

- **General Ledger (GL) Division** — GL divisions typically comprise individual entities (e.g., companies) in your general ledger. You must set up a GL division for each such entity. The GL division's sole purpose in the system is to define the accounting period associated with financial transactions linked to contracts associated with the GL division (contracts are associated with GL divisions via their contract type). The system cares about accounting periods in order to prevent a user from booking moneys to closed periods. It also uses accounting periods when it produces the flat file that contains the consolidated journal entry that is interfaced to your general ledger (refer to *The GL Interface* for more information).

Note: When determining how many GL Divisions you need, be sure to consider your general ledger and how your chart-of-accounts is structured. You will typically have one GL division for each "company" in your general ledger.

- **Division** — A division is typically associated with a jurisdiction. The definition of a jurisdiction is a geographic-oriented entity with unique business rules. For example, if you conduct business in California and Nevada, and each state has different collection rules, you will need a separate jurisdiction for each state. You must set up a division for each jurisdiction in which you conduct business.

Fastpath: Refer to *Setting Up Divisions* for information about divisions.

To define a general ledger division, select **Admin Menu, General Ledger Division**.

Description of Page

Enter a unique **GL Division** for the general ledger division.

Enter a **Description** of this general ledger division.

Define the accounting **Calendar ID** that controls how to convert an FT's accounting date into an accounting period. Refer to *Defining Accounting Calendars* for more information.

You may define a **Currency Code** for the GL division. Note that the system does not use this currency code.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_GL_DIVISION*.

Bank

A customer may have bank accounts in one or more banks. Oracle Revenue Management and Billing enables you to maintain banks and their bank accounts in the system. A tender source is then linked to a bank account from where the tender, such as cash, check, automatic payment, and so on is released. Each bank account can be linked to one or more tender source.

Bank (Used for Searching)

The **Bank** screen allows you to search for a bank using various search criteria. It also allows you to define a bank. It contains the following zone:

- *Search Bank* on page 60

Through this screen, you can navigate to the following screen:

- *Bank (Used for Viewing)* on page 62

Search Bank

The **Search Bank** zone allows you to search for a bank using various search criteria. It contains the following two sections:

- **Search Criteria** - The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Bank Code	Used to search a particular bank.	No
Description	Used to search banks with a particular description.	No

Note: You must specify at least one search criterion while searching for a bank.

- **Search Results** - On clicking the **Search** button, the search results appear based on the specified search criteria. The **Search Results** section contains the following columns:

Column Name	Column Description
Bank Code	Displays the bank code.
Description	Displays the description of the bank. **Note:** It has a link. On clicking the link, the **Bank** screen appears where you can view the details of the respective bank.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61

For more information on...	See...
How to view the details of a bank	*Viewing the Bank Details* on page 61
How to define a bank	*Defining a Bank* on page 62

Searching for a Bank

Procedure

To search for a bank:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **B** and then click **Bank**.

   The **Bank** screen appears.

3. Enter the search criteria in the **Search Bank** zone.

   Note: ORMB search engine supports wildcard search, where you can substitute the percentage (%) symbol as a stand in for any word or letter in a search criteria. You can use the '%' wildcard character in all input fields except the date and ID fields. The '%' wildcard character is suffixed automatically at the end of the partial search criteria. Therefore, you may or may not specify the wildcard character at the end of the partial search criteria. However, you have to prefix the wildcard character manually wherever required.

4. Click **Search**.

   A list of banks that meet the search criteria appears in the **Search Results** section.

Related Topics

For more information on...	See...
Bank screen	*Bank (Used for Searching)* on page 60
Search Bank zone	*Search Bank* on page 60

Viewing the Bank Details

Procedure

To view the details of a bank:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to view.

   The **Bank** screen appears.

3. View the details of the bank in the **Bank** zone.
4. View the list of accounts offered by the bank to the customer in the **Bank Accounts** zone.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62

For more information on...	See...
Bank zone	*Bank* on page 62
Bank Accounts zone	*Bank Accounts* on page 63

Defining a Bank

Procedure

To define a bank:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.
2. From the **Admin** menu, select **B** and then click **Bank**.

   The **Bank** screen appears.
3. Click the **Add** button in the **Page Title** area of the **Bank** screen.

   The **Bank** screen appears. It contains one section named **Main** which has the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Bank Code	Used to specify the bank code.	Yes
Description	Used to specify the description for the bank.	Yes

4. Enter the required details in the **Main** section.
5. Click **Save**.

   The bank is defined.

Related Topics

For more information on...	See...
Bank screen	*Bank (Used for Searching)* on page 60

Bank (Used for Viewing)

The **Bank** screen allows you to view the details of a bank including its bank accounts. In addition, you can:

- Edit the details of a bank and its bank accounts
- Delete and copy a bank
- Delete a bank account
- Define characteristics for each bank account

This screen contains the following zones:

- *Bank* on page 62
- *Bank Accounts* on page 63
- *Bank Account* on page 64
- *Bank Account Characteristics* on page 65

Bank

The **Bank** zone displays the details of the bank. It contains the following sections:

- **Main** - Displays basic information about the bank. It contains the following fields:

Field Name	Field Description
Bank Code	Displays the bank code.
Description	Displays the description of the bank.

- **Record Actions** - This section contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the bank.
Delete	Used to delete the bank. **Note:** You can only delete a bank whose bank accounts are not yet associated to a tender source.
Duplicate	Used to create a new bank using an existing bank.

- **Record Information** - This section contains the following field:

Field Name	Field Description
Business Object	Indicates the business object using which the bank is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.

Related Topics

For more information on...	See...
How to edit a bank	*Editing a Bank* on page 65
How to delete a bank	*Deleting a Bank* on page 66
How to copy a bank	*Copying a Bank* on page 66

Bank Accounts

The **Bank Accounts** zone lists the accounts offered by the bank to the customer. It contains the following columns:

Column Name	Column Description
Bank Account	Displays the type of the bank account.
Description	Displays the description of the bank account.
Account Number	Displays the number that identifies the bank account.
Currency	Indicates the currency in which the bank account is denominated.
Distribution Code	Indicates the distribution code, which in turn indicates the GL account associated with the bank account.
Edit	On clicking the **Edit** (✎) icon, the **Bank Account** screen appears where you can edit the details of the bank account.

Column Name	Column Description
Delete	On clicking the **Delete** ( ) icon, you can delete the bank account. **Note:** You can only delete a bank account which is not yet associated to a tender source.

You can define a bank account by clicking the **Add** link in the upper right corner of this zone. On clicking the **Broadcast** ( ) icon corresponding to the bank account, the **Bank Account** and **Bank Account Characteristics** zones appear.

Related Topics

For more information on...	See...
How to view the details of a bank account	*Viewing the Bank Account Details* on page 71
How to view the characteristics of a bank account	*Viewing Characteristics of a Bank Account* on page 71
How to define a bank account	*Defining a Bank Account* on page 67
How to edit a bank account	*Editing a Bank Account* on page 69
How to delete a bank account	*Deleting a Bank Account* on page 70

Bank Account

The **Bank Account** zone displays the details of the bank account. It contains the following sections:

- **Main** - Displays basic information about the bank account. It contains the following fields:

Field Name	Field Description
Bank Account	Indicates the type of the bank account.
Description	Displays the description of the bank account.
Account Number	Displays the number that identifies the bank account.
Check Digit	Displays the number that is located on the far right side of a bar code.
Branch ID	Indicates the branch in which the bank account is held.
Currency	Indicates the currency in which the bank account is denominated.
DFI ID	Indicates the depository financial institution for the bank account.
Distribution Code	Indicates the distribution code, which in turn indicates the GL account associated with the bank account.

- **Record Actions** - This section contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the bank account.
Delete	Used to delete the bank account. **Note:** You can only delete a bank account which is not yet associated to a tender source.

By default, the **Bank Account** zone does not appear in the **Bank** screen. It appears only when you click the **Broadcast** () icon corresponding to the bank account in the **Bank Accounts** zone.

Related Topics

For more information on...	See...
How to edit a bank account	*Editing a Bank Account* on page 69
How to delete a bank account	*Deleting a Bank Account* on page 70

Bank Account Characteristics

The **Bank Account Characteristics** zone lists the characteristics defined for the bank account. It contains the following columns:

Column Name	Column Description
Effective Date	Displays the date from when the characteristic is effective for the bank account.
Characteristic Type	Displays the characteristic type.
Characteristic Value	Displays the value of the characteristic type.

By default, the **Bank Account Characteristics** zone does not appear in the **Bank** screen. It appears only when you click the **Broadcast** () icon corresponding to the bank account in the **Bank Accounts** zone.

Editing a Bank

Procedure

To edit a bank:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to edit.

   The **Bank** screen appears.
3. Click the **Edit** button in the **Bank** zone.

   The **Bank** screen appears. It contains one section named **Main** which has the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Bank Code	Displays the bank code.	Not applicable
Description	Used to specify the description for the bank.	Yes
4. Modify the required details in the **Main** section.
5. Click **Save**.

   The changes made to the bank are saved.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank zone	*Bank* on page 62

Copying a Bank

Instead of creating a bank from scratch, you can create a new bank using an existing bank. This is possible through copying a bank. On copying a bank, the bank accounts are also copied to the new bank. You can then edit the details, if required.

Prerequisites

To copy a bank, you should have:

- Bank (whose copy you want to create) defined in the application

Procedure

To copy a bank:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose copy you want to create.

   The **Bank** screen appears.
3. Click the **Duplicate** button in the **Bank** zone.

   The **Bank** screen appears. It contains one section named **Main** which has the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Bank Code	Used to specify the bank code.	Yes
Description	Used to specify the description for the bank.	Yes

4. Enter the required details in the **Main** section.
5. Click **Save**.

   The new bank is defined.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank zone	*Bank* on page 62

Deleting a Bank

Procedure

To delete a bank:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank that you want to delete.

   The **Bank** screen appears.
3. Click the **Delete** button in the **Bank** zone.

   A message appears confirming whether you want to delete the bank.

Note: You can only delete a bank whose bank accounts are not yet associated to a tender source.

4. Click **OK**.

   The bank is deleted.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank zone	*Bank* on page 62

Defining a Bank Account

Prerequisites

To define a bank account, you should have:

- Currencies and distribution codes defined in the application

Procedure

To define a bank account:

1. Search for the bank, in which you want to add a bank account, in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to edit.

   The **Bank** screen appears.
3. Click the**Add** link in the upper right corner of the **Bank Accounts** zone.

   The **Bank Account** screen appears. It contains the following sections:

   - **Main** - Used to specify the basic details about the bank account.
   - **Characteristics** - Used to define characteristics for the bank account.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Bank Code	Indicates the bank for which you want to define the bank account.	Not applicable
Bank Account	Used to specify the type of the bank account.	Yes
Description	Used to specify the description for the bank account.	Yes
Account Number	Used to specify the number that identifies the bank account.	No
Check Digit	Used to specify the number that is located on the far right side of a bar code.	No
Branch ID	Used to indicate the branch in which the bank account is held.	No
Currency	Used to indicate the currency in which the bank account is denominated. **Note:** By default, the currency specified in the **Main** tab of the **Installation Options - Framework** screen appears in this field.	Yes
DFI ID	Used to indicate the depository financial institution for the bank account.	No

Field Name	Field Description	Mandatory (Yes or No)
Distribution Code	Used to indicate the distribution code, which in turn indicates the GL account associated with the bank account. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Distribution Code Search** window appears.	Yes

4. Enter the required details in the **Main** section.
5. Define characteristics for the bank account, if required.
6. Do either of the following:

If you want to...	Then
Save the bank account and exit from the screen	Click **Save**. The bank account is defined and listed in the **Bank Accounts** zone.
Save the bank account and then add another bank account	Click **Save and Add New**. The bank account is defined. All fields in the **Bank Account** screen are cleared, thereby allowing you to add another bank account.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank Accounts zone	*Bank Accounts* on page 63

Defining Characteristics for a Bank Account

Prerequisites

To define characteristics for a bank account, you should have:

- Characteristic types defined in the application (where the characteristic entity is set to **Bank Account**)

Procedure

To define characteristics for a bank account:

1. Ensure that the **Characteristics** section is expanded when you are defining or editing a bank account.

   The **Characteristics** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Effective Date	Used to specify the date from when the characteristic is effective for the bank account.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the bank account.

Field Name	Field Description	Mandatory (Yes or No)
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types where the characteristic entity is set to **Bank Account**.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the bank account.
Characteristic Value	Used to specify the value for the characteristic type. **Note:** On specifying the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the bank account.

2. Enter the required details in the **Characteristics** section.

Note: If you select a predefined characteristic type, the **Search** ( ) icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, you can search for a predefined characteristic value.

3. If you want to define more than one characteristic for the bank account, click the **Add** ( ) icon and then repeat step 2.

Note: However, if you want to remove a characteristic from the bank account, click the **Delete** ( ) icon corresponding to the characteristic.

Related Topics

For more information on...	See...
How to define a bank account	*Defining a Bank Account* on page 67
How to edit a bank account	*Editing a Bank Account* on page 69

Editing a Bank Account

Prerequisites

To edit a bank account, you should have:

- Currencies and distribution codes defined in the application

Procedure

To edit a bank account:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to edit.

   The **Bank** screen appears.
3. In the **Bank Accounts** zone, click the **Edit** icon ( ) in the **Edit** column corresponding to the bank account whose details you want to edit.

   The **Bank Account** screen appears. It contains the following sections:

   - **Main** - Used to specify the basic details about the bank account.
   - **Characteristics** - Used to define characteristics for the bank account.

The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Bank Code	Indicates the bank whose the bank account you are editing.	Not applicable
Bank Account	Indicates the type of the bank account.	Not applicable
Description	Used to specify the description for the bank account.	Yes
Account Number	Used to specify the number that identifies the bank account.	No
Check Digit	Used to specify the number that is located on the far right side of a bar code.	No
Branch ID	Used to indicate the branch in which the bank account is held.	No
Currency	Used to indicate the currency in which the bank account is denominated.	Yes
DFI ID	Used to indicate the depository financial institution for the bank account.	No
Distribution Code	Used to indicate the distribution code, which in turn indicates the GL account associated with the bank account. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Distribution Code Search** window appears.	Yes

Tip: Alternatively, you can edit the details of a bank account by clicking the **Edit** button in the **Bank Account** zone.

4. Modify the required details in the **Main** section.
5. Define, edit, or remove characteristics from the bank account, if required.
6. Click **Save**.

   The changes made to the bank account are saved.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank Accounts zone	*Bank Accounts* on page 63
Bank Account zone	*Bank Account* on page 64
How to define characteristics for a bank account	*Defining Characteristics for a Bank Account* on page 68

Deleting a Bank Account

Procedure

To delete a bank account:

1. Search for the bank in the **Bank** screen.

2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to edit.

   The **Bank** screen appears.

3. In the **Bank Accounts** zone, click the **Delete** ( ) icon in the **Delete** column corresponding to the bank account that you want to delete.

   A message appears confirming whether you want to delete the bank account.

   Note: You can only delete a bank account which is not yet associated to a tender source.

   Tip: Alternatively, you can delete a bank account by clicking the **Delete** button in the **Bank Account** zone.

4. Click **OK**.

   The bank account is deleted.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank Accounts zone	*Bank Accounts* on page 63
Bank Account zone	*Bank Account* on page 64

Viewing the Bank Account Details

Procedure

To view the details of a bank account:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to view.

   The **Bank** screen appears.

3. In the **Bank Accounts** zone, click the

   Broadcast ( ) icon corresponding to the bank account whose details you want to view.

   The **Bank Account** and **Bank Account Characteristics** zones appear.

4. View the details of the bank account in the **Bank Account** zone.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank Accounts zone	*Bank Accounts* on page 63
Bank Account zone	*Bank Account* on page 64

Viewing Characteristics of a Bank Account

Procedure

To view the characteristics of a bank account:

1. Search for the bank in the **Bank** screen.
2. In the **Search Results** section, click the link in the **Description** column corresponding to the bank whose details you want to view.

   The **Bank** screen appears.
3. In the **Bank Accounts** zone, click the

   Broadcast () icon corresponding to the bank account whose details you want to view.

   The **Bank Account** and **Bank Account Characteristics** zones appear.
4. View the characteristics of the bank account in the **Bank Account Characteristics** zone.

Related Topics

For more information on...	See...
How to search for a bank	*Searching for a Bank* on page 61
Bank screen	*Bank (Used for Viewing)* on page 62
Bank Accounts zone	*Bank Accounts* on page 63
Bank Account Characteristics zone	*Bank Account Characteristics* on page 65

Setting Up Service Types

You will have one service type for each type of service you provide to your customers.

Service Type - Main

To define service types and the types of facility levels, select **Admin Menu, Service Type**.

Description of Page

Enter a unique **Service Type** and **Description** for each service type.

Use the **Facility Level Names** collection to define the **Facility Level** and **Description** for each level in the service type's hierarchy.

Move to the **Level 1** tab to maintain the valid values for the highest facility level.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SVC_TYPE*.

Service Type - Level 1

Open **Admin Menu, Service Type** and navigate to the **Level 1** tab to define the various facilities classified at the highest level.

Description of Page

You can optionally start the grid at a given **Facility Level 1**.

Enter a **Facility Level 1** code and a **Description** for each facility in the highest level.

Service Type - Level 2

Open **Admin, Service Type** and navigate to the **Level 2** tab to define the various facilities classified at the second level.

Description of Page

You can optionally start the grid at a given **Facility Level 2**.

Enter a **Facility Level 2** code and a **Description** for each facility at the second level.

Service Type - Level 3

Open **Admin Menu, Service Type** and navigate to the **Level 3** tab to define the various facilities classified at the third level.

Description of Page

You can optionally start the grid at a given **Facility Level 3**.

Enter a **Facility Level 3** code and a **Description** for each facility at the third level.

To Do Lists Addendum

This section is an addendum to the general *To Do Lists* chapter. This addendum describes the To Do functionality that is specific to Oracle Revenue Management and Billing.

Assigning A To Do Role

As described in *To Do Entries Reference A Role*, each To Do entry requires a role. To Do entries created in Oracle Revenue Management and Billing may attempt to assign a role based on an account management group or division if it is applicable to the type of data related to the To Do entry.

As described in *The Big Picture of To Do Lists*, users are informed that something requires their attention by entries that appear in a To Do List.

You can optionally use account management groups (AMG) to define the respective role to be assigned to To Do entries that are associated with an account and To Do type. For example, you can create an AMG called `Credit Risks` and assign this to accounts with suspect credit. Then, whenever an account-oriented To Do entry is created for such an account, it will be assigned a role based on the `Credit Risks` AMG. Refer to *Setting Up Account Management Groups* for more information.

By assigning an AMG to an account, you are telling the system to address this account's To Do list entries to the roles defined on the AMG (note, each To Do type can have a different role defined for it on an AMG).

You can optionally use division to define the respective role to be assigned to To Do entries that are associated with an account and To Do type. For example, you may have a division called `California Operations` and assign this to accounts located in California. Then, whenever an account-oriented To Do entry is created for such an account, it will be assigned a role based on the `California Operations` division. Refer to *Setting Up Divisions* for more information.

A `To Do Pre-Creation` installation options plug-in is provided to determine the appropriate To Do Role for an account based on AMG and division setup. If plugged in, the logic to determine To Do role for an account is performed whenever a To Do entry is created. Refer to *C1-TDCR-DFRL* for further details on how this plug-in works.

Fastpath: Refer to *To Do Entries Reference A Role* for the details of how an initial role is assigned to To Do entries.

System To Do Types

Note: List of available To Do types. The To Do types available with the product may be viewed in the *application viewer*'s *To Do type* viewer. In addition if your implementation adds To Do types, you may *regenerate* the application viewer to see your additions reflected there.

Audit Trail Summary

The **Audit Trail Summary** screen allows you to view the audit trail information of fields for which audit is enabled on various user actions, such as add, update, and/or delete. This screen consists of the following zones:

Search

The **Search** zone allows you to search for tables where audit is enabled for all or some of the fields. This zone contains the following two sections:

- **Search Criteria** — The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Audited Table Name	Used to specify the table name.	No
Audited Table Description	Used to specify the description of the table.	No

- **Search Results** — On clicking the **Refresh** button, the search results appear based on the specified search criteria. The **Search Results** section contains the following columns:

Column Name	Column Description
Audited Table Name	Displays the table name.
Audited Table Description	Displays the description of the table.

Related Topics

For more information on...	See...
How to search for an audited table	*Searching for an Audited Table* on page 75

Audited Fields

The **Audited Fields** zone lists the audit trail information of fields for which audit is enabled on various user actions, such as add, update, and/or delete. You can filter the list using various search criteria. This zone contains the following two sections:

- **Search Criteria** — The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Audited Field Name	Used to indicate the field whose audit trial information you want to view.	No

Field Name	Field Description	Mandatory (Yes or No)
<Primary Key Field>	Used to indicate the record whose audit trial information you want to view. **Note:** The primary key dynamically changes depending on the table that you have selected. For example, if you have clicked the **Broadcast** icon corresponding to the CI_ACCT_APAY table, the ACCT_APAY_ID field appears as the primary key. However, if a combination of two or more fields is used as a primary key for a table, then all those fields appear in this section.	No
Creation Start Date/Time	Used to search for all changes which were made from a particular date and time onwards.	No
Creation End Date/Time	Used to search for all changes which were made till a particular date and time.	No

- **Search Results** — On clicking the **Refresh** button, the search results appear based on the specified search criteria. The **Search Results** section contains the following columns:

Column Name	Column Description
Create Date/Time	Displays the date and time when the change was made.
User Name	Indicates the user who has made the change.
Primary Key	Indicates the record in which the change was made
Audited Field Name	Indicates the field for which the change was made.
Audit Action	Indicates the action performed by the user. The valid values are: • Add • Update • Delete
Value Before Update	Displays the value of the field before the user action.
Value After Update	Displays the value of the field after the user action.

Note: By default, the **Audited Fields** zone does not appear in the **Audit Trail Summary** screen. It appears only when you click the **Broadcast** ( ) icon corresponding to the audited table in the **Search Results** section.

Related Topics

For more information on...	See...
How to view the audit trail information of a table	*Viewing Audit Trail Information of a Table* on page 76

Searching for an Audited Table

Procedure

To search for an audited table:

1. Click the **Menu** link in the **Actions/Navigation** area.

   A list appears.
2. Select the **Admin Menu** option from the list.
3. From the **Admin Menu**, select **A** and then click **Audit Trail Summary**.

   The **Audit Trail Summary** screen appears.
4. Enter the search criteria in the **Search** zone.

   Note: ORMB search engine supports wildcard search, where you can substitute the percentage (%) symbol as a stand in for any word or letter in a search criteria. You can use the '%' wildcard character in all input fields except the date and ID fields. The '%' wildcard character is suffixed automatically at the end of the partial search criteria. Therefore, you may or may not specify the wildcard character at the end of the partial search criteria. However, you have to prefix the wildcard character manually wherever required.
5. Click **Refresh**.

   A list of tables (where audit is enabled and) that meet the search criteria appears in the **Search Results** section.
6. If required, you can export the list of audited tables in the Excel format by clicking the **Export To Excel** menu option from the **Explorer Zone** (⚙) menu.

Related Topics

For more information on...	See...
Audit Trail Summary screen	*Audit Trail Summary* on page 74
Search zone	*Search* on page 74

Viewing Audit Trail Information of a Table

Procedure

To view the audit trail information of a table:

1. Search for the table (where audit is enabled) in the **Audit Trail Summary** screen.
2. In the **Search Results** section, click the **Broadcast** (📡) icon corresponding to the table whose audit trail information you want to view.

   The **Audited Fields** zone appears.
3. View the audit trail information of the table in the **Audited Fields** zone.
4. If required, you can filter the audit trail information of the table.
5. If required, you can export the audit trail information in the Excel format by clicking the **Export To Excel** menu option from the **Explorer Zone** (⚙) menu.

Related Topics

For more information on...	See...
How to search for an audited table	*Searching for an Audited Table* on page 75
Audited Fields zone	*Audited Fields* on page 74

State

The **State** screen allows you to define, edit, and delete a state. It contains the following zones:

- *Search State* on page 77
- *State* on page 78

Search State

The **Search State** zone allows you to search for a state using various search criteria. It contains the following sections:

- **Search Criteria** - The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Country	Used to search states which belong to a particular country. **Note:** The list includes only those countries which are defined in the system.	Yes
State	Used to search a particular state. **Note:** The values appear in this list only when the country is selected. The list includes only those states which are defined for the country.	No

- **Search Results** - On clicking the **Search** button, the search results appear based on the specified search criteria. It contains the following columns:

Column Name	Column Description
Country	Displays the abbreviation of the country.
Country Description	Displays the country name.
State	Displays the abbreviation of the state.
State Description	Displays the state name.
Edit	On clicking the **Edit** ( ) icon, the **State** screen appears where you can edit the details of the state.
Delete	On clicking the **Delete** ( ) icon, you can delete the state. **Note:** You can delete a state only when it is not yet used in the system.

On clicking the **Broadcast** ( ) icon corresponding to the state, the **State** zone appears with the details of the respective state.

Related Topics

For more information on...	See...
How to search for a state	*Searching for a State* on page 78

For more information on...	See...
How to view the details of a state	*Viewing the State Details* on page 79
How to edit a state	*Editing a State* on page 81
How to delete a sate	*Deleting a State* on page 82

State

The **State** zone displays the details of the state. It contains the following sections:

- **Main** - Displays basic information about the state. It contains the following columns:

Field Name	Field Description
Country	Displays the abbreviation of the country.
Country Description	Displays the country name.
State	Displays the abbreviation of the state.
State Description	Displays the state name.

- **Characteristics** - Lists the characteristics defined for the state. It contains the following columns:

Column Name	Column Description
Characteristic Type	Displays the characteristic type.
Characteristic Value	Displays the value of the characteristic type.

Searching for a State

Prerequisites

To search for a state, you should have:

- Countries defined in the application

Procedure

To search for a state:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **S** and then click **State**.

   The **State** screen appears.

3. Enter the search criteria in the **Search State** zone.

   Note: ORMB search engine supports wildcard search, where you can substitute the percentage (%) symbol as a stand in for any word or letter in a search criteria. You can use the '%' wildcard character in all input fields except the date and ID fields. The '%' wildcard character is suffixed automatically at the end of the partial search criteria. Therefore, you may or may not specify the wildcard character at the end of the partial search criteria. However, you have to prefix the wildcard character manually wherever required.

4. Click **Search**.

   A list of states that meet the search criteria appears in the **Search Results** section.

Related Topics

For more information on...	See...
State screen	*State* on page 77
Search State zone	*Search State* on page 77

Viewing the State Details

Procedure

To view the details of a state:

1. Search for the state in the **State** screen.
2. In the **Search Results** section, click the **Broadcast** ( ) corresponding to the state whose details you want to view.

   The **State** zone appears.
3. View the details of the state in the **State** zone.

Related Topics

For more information on...	See...
How to search for a state	*Searching for a State* on page 78
State zone	*State* on page 78

Defining a State

Prerequisites

To define a state, you should have:

- Countries defined in the application

Procedure

To define a state:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.
2. From the **Admin** menu, select **S** and then click **State**.

   The **State** screen appears.
3. Click the **Add** button in the **Page Title** area of the **State** screen.

   The **State** screen appears. It contains the following sections:

   - **Main** – Used to specify basic information about the state.
   - **Characteristics** – Used to define characteristics for the state.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Country	Used to indicate the country for which you want to define a state.	Yes
State	Used to specify an abbreviation for the state. For example, **CA** for California.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Description	Used to specify the state name. For example, **California**.	Yes

4. Enter the required details in the **Main** section.
5. Define characteristics for the state, if required.
6. Click **Save**.

   The state is defined.

Related Topics

For more information on...	See...
State screen	*State* on page 77
How to define a characteristic for a state	*Defining a Characteristic for a State* on page 80

Defining a Characteristic for a State

Prerequisites

To define a characteristic for a state, you should have:

- Characteristic types defined in the application (where the characteristic entity is set to **State**).

Procedure

To define a characteristic for a state:

1. Ensure that the **Characteristics** section is expanded when you are defining or editing a state.

   The **Characteristics** section contains the following fields in the grid:

Field Name	Field Description	Mandatory (Yes or No)
Effective Date	Used to specify the date from when the characteristic is effective for the state.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the state.
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types where the characteristic entity is set to **State**.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the state.

Field Name	Field Description	Mandatory (Yes or No)
Characteristic Value	Used to specify the value for the characteristic type. **Note:** If you select a predefined characteristic type, the **Search** ( ) icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, the **Predefined Characteristic Search** window appears. On specifying the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the state.

2. Enter the required details in the **Characteristics** section.
3. If you want to define more than one characteristic for the state, click the **Add** ( ) icon and then repeat step 2.

   Note: However, if you want to remove a characteristic from the state, click the **Delete** ( ) icon corresponding to the characteristic.
4. Click **Save**.

   The characteristics are defined for the state.

Related Topics

For more information on...	See...
How to define a state	*Defining a State* on page 79
How to edit a state	*Editing a State* on page 81

Editing a State

Procedure

To edit a state:

1. Search for the state in the **State** screen.
2. In the **Search Results** section, click the **Edit** ( ) icon corresponding to the state whose details you want to edit.

   The **State** screen appears. It contains the following sections:

   - **Main** – Used to specify basic information about the state.
   - **Characteristics** – Used to define characteristics for the state.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Country	Displays the country name.	Not applicable
State	Displays the abbreviation of the state.	Not applicable
Description	Used to specify the state name.	Yes

3. Modify the required details in the **Main** section.
4. Define, edit, or remove characteristics from the state, if required.
5. Click **Save**.

   The changes made to the state are saved.

Related Topics

For more information on...	See...
How to search for a state	*Searching for a State* on page 78
How to define a characteristic for a state	*Defining a Characteristic for a State* on page 80

Deleting a State

Procedure

To delete a state:

1. Search for the state in the **State** screen.
2. In the **Search Results** section, click the **Delete** ( ) icon corresponding to the state that you want to delete.

   A message appears confirming whether you want to delete the state.

   Note: You can delete a state only when it is not yet used in the system.
3. Click **OK**.

   The state is deleted.

Related Topics

For more information on...	See...
How to search for a state	*Searching for a State* on page 78

Entity Audit

Oracle Revenue Management and Billing provides a mechanism wherein auditors can track various actions, such as add, update, and delete for an entity. If the entity audit feature is enabled for a business object, the system creates audit events whenever you define, edit, or delete an entity (which is created using the respective business object). For example, if the entity audit feature is enabled for the **C1-Membership** business object, the system creates an audit event whenever you define, edit, or delete a membership in a policy plan or whenever you add or remove persons from a membership. For more information on how to enable the entity audit feature for a business object, see *Prerequisites* on page 83.

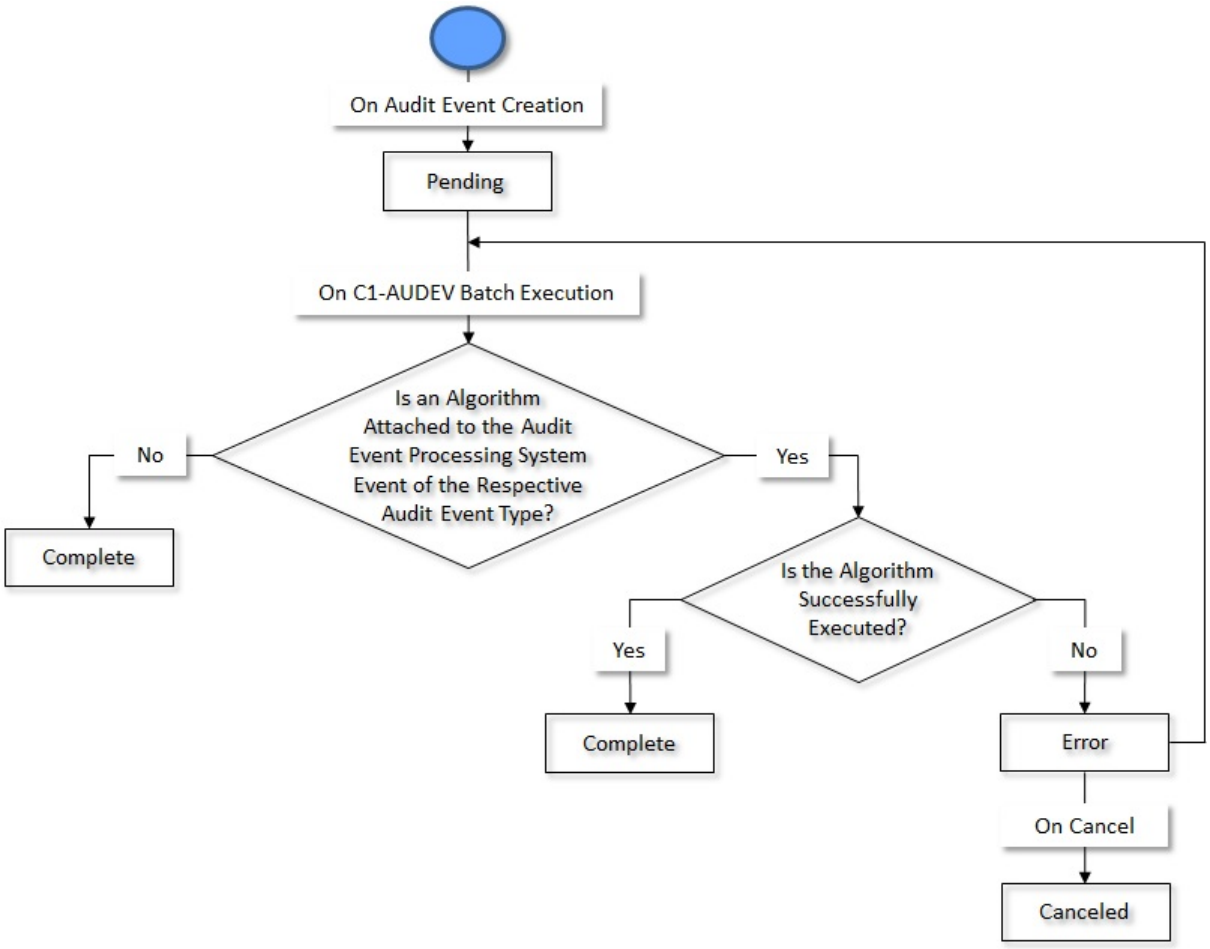
The entity audit process is a two-step process, wherein the system creates the audit events and then processes them whenever the **Audit Event Processing (C1-AUDEV)** batch is executed. For more information about these processes, see the following sections:

- *Audit Event Creation* on page 84
- *Audit Event Processing* on page 85

During the entity audit process, each audit event goes through various statuses in its lifecycle. For more information about the audit event statuses, see *Audit Event Status Transition* on page 83.

Audit Event Status Transition

The following figure graphically indicates how an audit event moves from one status to another in its lifecycle:



Prerequisites

To setup the entity audit feature for a business object, you need to do the following:

- Set the **Eligible for Audit Event** option type of the entity business object to **Y**.

  Note: The **Eligible for Audit Event** option type is available for the entity business object only when the **Valid BO Option Type** option type of the respective maintenance object is set to **C1AT**.

- Attach an algorithm created using the **C1-REAUDEVNT** algorithm type to the **Audit** system event of the entity business object.

Note:

At present, this algorithm is designed to work with the following business objects:

- C1-POLICY
- C1-ASOPolicy
- C1-Membership
- C1-PolicyPlan
- C1_PERSON_BO
- C1-PricingRuleTierBased
- C1-PricingRuleAgeBased

If you want to use the audit algorithm for the business objects which are not listed above, you need to create a custom audit algorithm.

- Define an active audit event type for the entity business object in the system.
- Define the required algorithm types where the algorithm entity is set to **Audit Event Processing**.
- Create the required algorithms and attach them to the **Audit Event Processing** system event of the audit event type.

Note: The **C1-AUDEVMPR** algorithm type is shipped with the product. You must attach an algorithm created using the **C1-AUDEVMPR** algorithm type to the audit event types which are defined for the **C1-PricingRuleTierBased** and **C1-PricingRuleAgeBased** business objects.

- Define the values for the **AUDIT_USAGE_FLG** lookup field
- Define the required characteristic types where the characteristic entity is set to **Audit Event Type**.

Entity Audit Process

The entity audit process consists of the following two sub-processes:

- *Audit Event Creation* on page 84
- *Audit Event Processing* on page 85

Audit Event Creation

If you add, edit, or delete an entity which is created using a business object for which the entity audit feature is enabled, the system checks whether an active audit event type exists for the entity business object. If so, it considers the active audit event type and creates the audit event using the respective audit event type. The system creates the audit event while:

- Defining an entity when the **Add Action** option is selected in the audit event type
- Editing an entity when the **Update All** option is selected in the audit event type or when the updated element is listed in the **Audit Elements for Entity Update** section
- Deleting an entity when the **Delete Action** option is selected in the audit event type

For more information about the audit event type, see *Audit Event Type* on page 86.

The audit event is created in the **Pending** status. Note that the system creates one audit event for the entity irrespective of the number of changes made during the entity update. The entity type and entity ID for which an audit event is created are added corresponding to the audit event in the **C1_AUDIT_EVENT** table. For example, if the audit event is created while adding or updating a pricing rule, then the entity type is set to **Pricing Rule** and entity ID is set to the pricing rule ID. In addition, an effective date is stamped corresponding to the audit event in the **C1_AUDIT_EVENT** table. Note

that if the entity has a start date, then the effective date is set to the entity's start date. But, if the entity does not have a start date, then the effective date is set to the system date.

At a time, you can have only two audit events for the entity in the **Pending** status — one audit event which is created during the **Add** action and another audit event which is created during the **Update** action.

Note: Before creating an audit event, the system checks whether an audit event for the entity ID with the same effective date already exists in the **Pending** or **Error** status for the respective action. If so, the system does not create a new audit event for the entity. Instead, the system adds a new log entry in the existing audit event.

Audit Event Processing

On executing the **Audit Event Processing (C1-AUDEV)** batch, the system checks whether there are any audit events in the **Pending** status. If there is an audit event in the **Pending** status, the system checks whether an algorithm is attached to the **Audit Event Processing** system event of the respective audit event type. If an algorithm is not attached to the **Audit Event Processing** system event of the respective audit event type, the status of the audit event is set to **Complete**.

However, if an algorithm is attached to the **Audit Event Processing** system event of the respective audit event type, the system executes the algorithm and accordingly changes the status of the audit event to **Complete**. For example, while processing the audit events of the **C1-PricingRuleTierBased** and **C1-PricingRuleAgeBased** business objects, the system checks whether an algorithm is attached to the **Audit Event Processing** system event of the respective audit event type. If the **C1-AUDEVMPR** algorithm is attached to the audit event type, the system identifies the policy plan where the pricing rule is defined. Once the policy plan is identified, it extracts a list of membership defined on the policy plan and the pricing rule type using which the pricing rule is created. Then, the system creates an entry for each membership, pricing rule type, and effective date combination in the **CI_REPRC_ENTITY_DTL** table. The status of these entries is set to **P**. Finally, the status of the audit event is set to **Complete**.

If any error occurs, the status of the audit event is set to **Error**. The system enables you to either reprocess or cancel the audit events which are in the **Error** status. On canceling an audit event, the status of the audit event is changed to **Canceled**.

For more information about the **Audit Event Processing (C1-AUDEV)** batch and its parameters, see *Oracle Revenue Management and Billing Batch Guide*.

Algorithms Used in the Entity Audit Process

The following table lists the algorithms which are designed to implement the entity audit process:

Algorithm	Algorithm Type	Attached To	System Event	Algorithm Description
C1-REAUDEVNT	C1-REAUDEVNT	Entity Business Object	Audit	Refer to *C1-REAUDEVNT* on page 85.
C1-AUDEVMPR	C1-AUDEVMPR	Audit Event Type of the **C1-PricingRuleAgeBased** and **C1-PricingRuleTierBased** business objects	Audit Event Processing	Refer to *C1-AUDEVMPR* on page 86.

C1-REAUDEVNT

If this algorithm is attached to the **Audit** system event of a business object, it is invoked whenever you define, edit, or delete the respective entity. It checks whether an active audit event type exists for the entity business object. If so, it considers the active audit event type and creates the audit event using the respective audit event type. The system creates the audit event while:

- Defining an entity when the **Add Action** option is selected in the audit event type
- Editing an entity when the **Update All** option is selected in the audit event type or when the updated element is listed in the **Audit Elements for Entity Update** section

- Deleting an entity when the **Delete Action** option is selected in the audit event type

The entity type and entity ID for which an audit event is created are added corresponding to the audit event in the **C1_AUDIT_EVENT** table. In addition, the effective date is stamped corresponding to the audit event in the **C1_AUDIT_EVENT** table. Note that if the entity has a start date, then the effective date is set to the entity's start date. But, if the entity does not have a start date, then the effective date is set to the system date.

Note: Before creating an audit event, the system checks whether an audit event for the entity ID with the same effective date already exists in the **Pending** or **Error** status for the respective action. If so, the system does not create a new audit event for the entity. Instead, the system adds a new log entry in the existing audit event.

At present, this algorithm is designed to work with the following business objects:

- C1-POLICY
- C1-ASOPolicy
- C1-Membership
- C1-PolicyPlan
- C1_PERSON_BO
- C1-PricingRuleTierBased
- C1-PricingRuleAgeBased

If you want to use the audit algorithm for the business objects which are not listed above, you need to create a custom audit algorithm.

It contains the following parameters:

- **Audit Event Business Object** – Used to specify the business object using which you want to create the audit event.
- **Audit Event Pending Status** – Used to specify the status in which you want to create the audit event. This parameter is also used for determining whether an audit event for the entity ID already exists in the system.
- **Audit Event Error Status** – Used to specify the status in which an audit event is transitioned when an error occurs while processing the audit event. This parameter is used for determining whether an audit event for the entity ID already exists in the system.

All the above parameters are mandatory.

C1-AUDEVMPR

You must attach this algorithm to the **Audit Event Processing** system event while creating an audit event type for the **C1-PricingRuleAgeBased** and **C1-PricingRuleTierBased** business objects.

This algorithm is invoked when the **Audit Event Processing (C1-AUDEV)** batch is executed. If this algorithm is attached to the **Audit Event Processing** system event of the respective audit event type, the system identifies the policy plan where the pricing rule is defined. Once the policy plan is identified, it extracts a list of membership defined on the policy plan and the pricing rule type using which the pricing rule is created. Then, the system creates an entry for each membership, pricing rule type, and effective date combination in the **CI_REPRC_ENTITY_DTL** table. The status of these entries is set to **P** in the **CI_REPRC_ENTITY_DTL** table.

Audit Event Type

Oracle Revenue Management and Billing enables you to define an audit event type for an entity business object. However, at a time, only one audit event type of the entity business object can be in the **Active** status. The system creates an audit event using the active audit event type whenever you define, edit, or delete the respective entity. The audit event type helps the system to determine:

- **Entity Business Object** – The business object for which you want to define the audit event type.

- **Audit Usage** – The usage indicates the purpose of auditing an entity which is created using the entity business object. For example, membership premium charges might change due to change in pricing and therefore you may want to audit the pricing rules for premium calculation.
- **Audit Actions** – The various options, such as **Add Action**, **Delete Action**, and **Update All** are available while defining an audit event type. The system creates an audit event while defining, editing or deleting an entity depending on whether the respective option is selected in the audit event type. For example, if you create the active audit event types for the following entity business objects where the below listed options are selected:

Audit Event Type	Entity Business Object	Add Action	Update All	Delete Action
AETY1	C1-POLICY	Y	N	N
AETY2	C1-ASOPolicy	Y	Y	Y

Then, the system will create an audit event using the **AETY1** audit event type when you define a fully-insured policy. But, the system will not create an audit event when you edit or delete a fully-insured policy. However, the system will create an audit event using the **AETY2** audit event type when you define, edit, or delete a self-funded policy.

- **Audit Elements During Update** – Alternatively, you can list the elements of the entity business object which should be monitored for the auditing purpose while editing an entity in the audit event type. You can list the elements of the entity business object only when the **Update All** option is not selected. The system enables you to monitor a set of the business object statuses, characteristic types, and fields and accordingly creates an audit event while editing an entity. For example, if you create the audit event types for the following entity business objects:

Audit Event Type	Entity Business Object	Update All	Audit Elements During Update
AETY3	C1_PERSON_BO	Y	Not applicable
AETY4	C1-PricingRuleTierBased	N	Pricing Start Date, Rate Option, Base Fee for a Price Component
AETY5	C1-PricingRuleAgeBased	N	Rate Option, Base Fee for a Price Component

Then, the system will create an audit event using the **AETY3** audit event type when you edit any details of a person. But, the system will create an audit event using the **AETY4** audit event type only when the following fields in a tier based pricing rule are edited:

  - Pricing Start Date
  - Rate Option
  - Base Fee for a Price Component

Similarly, the system will create an audit event using the **AETY5** audit event type only when the following fields in an age based pricing rule are edited:

  - Rate Option
  - Base Fee for a Price Component

The **Audit Event Type** screen allows you to define, edit, copy, and delete an audit event type. It contains the following zones:

- *Audit Event Type List* on page 88
- *Audit Event Type* on page 89

Audit Event Type List

The **Audit Event Type List** zone lists the audit event types that are already defined in the system. It contains the following columns:

Column Name	Column Description
Audit Event Type	Displays the audit event type.
Description	Displays the description of the audit event type.
Entity Business Object	Indicates the business object for which the audit event type is created. In addition, this column has a context menu which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.
Status	Indicates the status of the audit event type. The valid values are: • Active • Inactive
Audit Usage	Indicates the purpose of auditing the entity business object. The valid value is: • Premium Calculation
Edit	On clicking the **Edit** ( ) icon, the **Audit Event Type** screen appears where you can edit the details of the audit event type. **Note:** You can edit an audit event type only when an audit event is not yet created using the audit event type. However, if an audit event is created using the audit event type, you can only do the following: • Change the status from **Active** to **Inactive** and vice-versa • Attach algorithms (if required) to the audit event type • Define, edit, or remove characteristics from the audit event type
Duplicate	On clicking the **Duplicate** ( ) icon, the **Audit Event Type** screen appears where you can define a new audit event type using an existing audit event type.
Delete	On clicking the **Delete** ( ) icon, you can delete the audit event type. **Note:** You can delete an audit event type only when an audit event is not yet created using the audit event type.

Note: Pagination is used to display limited number of records in this zone. You can use the navigation links, such as **Previous** and **Next** to navigate between pages.

You can filter the list using various search criteria (such as, **Status** and **Entity Business Object**) available in the **Filter** area. By default, the **Filter** area is hidden. You can view the **Filter** area by clicking the **Filters** ( ) icon in the upper right corner of this zone.

On clicking the **Broadcast** ( ) icon corresponding to an audit event type, the **Audit Event Type** zone appears with the details of the respective audit event type.

Related Topics

For more information on...	See...
How to edit an audit event type	*Editing an Audit Event Type* on page 99
How to copy an audit event type	*Copying an Audit Event Type* on page 101
How to delete an audit event type	*Deleting an Audit Event Type* on page 103
How to view the details of an audit event type	*Viewing the Audit Event Type Details* on page 104

Audit Event Type

The **Audit Event Type** zone displays the details of the audit event type. It contains the following sections:

- **Main** - Displays basic information about the audit event type. It contains the following fields:

Field Name	Field Description
Audit Event Type	Displays the audit event type.
Description	Displays the description of the audit event type.
Detailed Description	Displays additional information about the audit event type.
Status	Indicates the status of the audit event type. The valid values are: • Active • Inactive
Entity Business Object	Indicates the business object for which the audit event type is defined. In addition, this field has a context menu which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.
Audit Usage	Indicates the purpose of auditing the entity business object. The valid value is: • Premium Calculation
Update All	Indicates whether an audit event must be created whenever any field, characteristic, or status of the entity is updated.
Add Action	Indicates whether an audit event must be created whenever an entity is added.
Delete Action	Indicate whether an audit event must be created whenever an entity is deleted.

- **Algorithms** - Lists the algorithms associated with the audit event type. It contains the following columns:

Column Name	Column Description
System Event	Indicates the system event when the algorithm must be triggered. The valid value is: • Audit Event Processing
Sequence	Indicates the order in which the algorithms with the same system event must be triggered.
Algorithm	Indicates the algorithm attached to the system event. **Note:** It has a link. On clicking the link, the **Algorithm** screen appears where you can view the details of the respective algorithm.

- **Audit Elements for Entity Update** - Lists the elements which must be monitored for auditing while updating the entity. It contains the following columns:

Column Name	Column Description
Element Type	Indicates the type of element. The valid values are: • Business Object Status • Characteristic • Field
Element Name	Displays the element which must be monitored for auditing. **Note:** If the element is a part of a group or list, the system displays the element name along with the element group and/or list. For example, if you select the **policyPerRole** element of the **C1-ASOPolicy** business object, the system displays the `policyPersons/policyPerRole` in the **Element Name** field. Here, the **policyPersons** is the element list.
Element Value	The following table lists the element value depending on the element type: **Element Type** / **Element Value** Business Object Status: Indicates the status of the entity business object which must be monitored for auditing. Characteristic: Indicates the characteristic type which must be monitored for auditing. **Note:** The data does not appear in this column when the element type is set to **Field**.

- **Characteristics** - Lists the characteristics which are defined for the audit event type. It contains the following columns:

Column Name	Column Description
Characteristic Type	Indicates the characteristic type.
Characteristic Value	Displays the value of the characteristic type.

- **Record Actions** - This section contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the audit event type. **Note:** You can edit an audit event type only when an audit event is not yet created using the audit event type. However, if an audit event is created using the audit event type, you can only do the following: • Change the status from **Active** to **Inactive** and vice-versa • Attach algorithms (if required) to the audit event type • Define, edit, or remove characteristics from the audit event type

Button Name	Button Description
Delete	Used to delete the audit event type. **Note:** You can delete an audit event type only when an audit event is not yet created using the audit event type.
Duplicate	Used to create a new audit event type using an existing audit event type.

- **Record Information** - This section contains the following field:

Field Name	Field Description
Business Object	Indicates the business object using which the audit event type is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.

By default, the **Audit Event Type** zone does not appear in the **Audit Event Type** screen. It appears only when you click the **Broadcast** ( ) icon corresponding to an audit event type in the **Audit Event Type List** zone.

Related Topics

For more information on...	See...
How to edit an audit event type	*Editing an Audit Event Type* on page 99
How to copy an audit event type	*Copying an Audit Event Type* on page 101
How to delete an audit event type	*Deleting an Audit Event Type* on page 103
How to view the details of an audit event type	*Viewing the Audit Event Type Details* on page 104

Defining an Audit Event Type

Prerequisites

To define an audit event type, you should have:

- Entity business objects defined in the application
- The **Eligible for Audit Event** option type of the entity business object (for which you want to define an audit event type) set to **Y**

  Note: The **Eligible for Audit Event** option type is available for the entity business object only when the **Valid BO Option Type** option type of the respective maintenance object is set to **C1AT**.

- Values defined for the **AUDIT_USAGE_FLG** lookup field

Procedure

To define an audit event type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Audit Event Type**.

   A sub-menu appears.

3. Click the **Add** option from the **Audit Event Type** sub-menu.

The **Audit Event Type** screen appears. It contains the following sections:

- **Main** - Used to specify basic information about the audit event type.
- **Algorithms** - Used to associate algorithms with the audit event type.
- **Audit Elements for Entity Update** - Used to list the elements from the entity business object which you want to monitor for auditing while updating the entity.

  Note: The **Audit Elements for Entity Update** section is enabled only when the **Update All** option is not selected in the **Main** section.

- **Characteristics** - Used to define characteristics for the audit event type.

The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Audit Event Type	Used to specify the audit event type. **Note:** You cannot specify any special characters except hyphen and underscore in the audit event type.	Yes
Business Object	Indicates the business object using which you are defining the audit event type.	Not applicable
Description	Used to specify the description for the audit event type.	Yes
Detailed Description	Used to specify additional information about the audit event type.	No
Status	Used to indicate the status of the audit event type. The valid values are: • Active • Inactive **Note:** You can create only one audit event type in the **Active** status for an entity business object.	Yes
Entity Business Object	Used to indicate the business object for which you want to define the audit event type. **Note:** The list includes only those business objects where the **Eligible for Audit Event** option type is set to **Y**.	Yes
Audit Usage	Used to indicate the purpose of auditing the entity business object. The valid value is: • Premium Calculation	Yes
Update All	Used to indicate whether you want to create an audit event whenever any field, characteristic, or status of the entity is updated.	No

Field Name	Field Description	Mandatory (Yes or No)
Add Action	Used to indicate whether you want to create an audit event whenever the entity is added.	Yes (Conditional) **Note:** This field is required when you want to create an audit event while adding the entity.
Delete Action	Used to indicate whether you want to create an audit event whenever the entity is deleted.	Yes (Conditional) **Note:** This field is required when you want to create an audit event while deleting the entity.

Tip: Alternatively, you can access this screen by clicking the **Add** button in the **Page Title** area of the **Audit Event Type** screen.

4. Enter the required details in the **Main** section.
5. Associate the required algorithms with the audit event type.
6. Add elements (such as, field, characteristic, or status) which you want to monitor for auditing while updating the entity.

   Note: You can list the elements for auditing only when the **Update All** option is not selected in the **Main** section.
7. Define characteristics for the audit event type, if required.
8. Click **Save**.

   The audit event type is defined.

Related Topics

For more information on...	See...
Audit Event Type screen	*Audit Event Type* on page 86
How to associate an algorithm with an audit event type	*Associating an Algorithm with an Audit Event Type* on page 93
How to add an element for auditing in an audit event type	*Adding an Element for Auditing in an Audit Event Type* on page 94
How to define a characteristic for an audit event type	*Defining a Characteristic for an Audit Event Type* on page 98

Associating an Algorithm with an Audit Event Type

Prerequisites

To associate an algorithm with an audit event type, you should have:

- Algorithms defined in the application (which are created using an algorithm type where the algorithm entity is set to **Audit Event Processing**)

  Note: The **C1-AUDEVMPR** algorithm type is shipped with the product. You must attach an algorithm created using the **C1-AUDEVMPR** algorithm type to the audit event types which are defined for the **C1-PricingRuleTierBased** and **C1-PricingRuleAgeBased** business objects.

Procedure

To associate an algorithm with an audit event type:

1. Ensure that the **Algorithms** section is expanded when you are defining, editing, or copying an audit event type.

   The **Algorithms** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
System Event	Used to indicate the system event when you want to trigger the algorithm. The valid value is: • **Audit Event Processing** - Used when you want to trigger the attached algorithm while executing the **Audit Event Process Monitor (C1-AUDEV)** batch.	Yes (Conditional) **Note:** This field is required when you are associating an algorithm with the audit event type.
Sequence	Used to indicate the order in which the algorithms with the same system event should be triggered.	Yes (Conditional) **Note:** This field is required when you are associating an algorithm with the audit event type.
Algorithm	Used to indicate the algorithm that you want to attach to the system event. **Note:** Once you select the system event, the **Search** () icon appears corresponding to the **Algorithm** field. On clicking the **Search** icon, the **Algorithm Search** window appears. On specifying the algorithm, the description of the algorithm appears corresponding to the **Algorithm** field.	Yes (Conditional) **Note:** This field is required when you are associating an algorithm with the audit event type.

2. Enter the required details in the **Algorithms** section.
3. If you want to associate more than one algorithm with the audit event type, click the **Add** () icon, and then repeat step 2.

Note: However, if you want to remove an algorithm from the audit event type, click the **Delete** () icon corresponding to the algorithm.

Related Topics

For more information on...	See...
How to define an audit event type	*Defining an Audit Event Type* on page 91
How to edit an audit event type	*Editing an Audit Event Type* on page 99
How to copy an audit event type	*Copying an Audit Event Type* on page 101

Adding an Element for Auditing in an Audit Event Type

Prerequisites

To add an element for auditing in an audit event type, you should have:

- Required elements defined in the entity business object schema
- Required statuses defined in the lifecycle of the entity business object

- Required characteristic types defined in the application (where the characteristic entity is set to the respective entity)

Procedure

To add an element for auditing in an audit event type:

1. Ensure that the **Audit Elements for Entity Update** section is expanded when you are defining, editing, *o*r copying an audit event type.

   The **Audit Elements for Entity Update** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Element Type	Used to indicate the type of element. The valid values are: • Business Object Status • Characteristic • Field	Yes (Conditional) **Note:** This field is required when you want to create an audit event whenever the specified element is updated.

<table>
<tr><th>Field Name</th><th>Field Description</th><th>Mandatory (Yes or No)</th></tr>
<tr><td>Element Name</td><td>Used to specify the element which you want to monitor while updating the entity. The following table lists the elements which you must select for the respective element type:

<table>
<tr><th>Element Type</th><th>Element Name</th></tr>
<tr><td>Business Object Status</td><td>boStatus</td></tr>
<tr><td>Characteristic</td><td>characteristics/characteristicsList/ characteristicType</td></tr>
<tr><td>Field</td><td>Any field from the entity business object schema

Note:
Here, you must not specify a element whose type is set to group or list.

If the element is a part of a group or list, the system displays the element name along with the element group and/or list. For example, if you select the policyPerRole element of the C1-ASOPolicy business object, the system displays the <code>policyPersons/ policyPerRole</code> in the Element Name field. Here, the policyPersons is the element list.</td></tr>
</table>

Note:
The Search () icon appears corresponding to the Element Name field. On clicking the Search icon, the Schema window appears.

The Schema window does not appear when the entity business object is not specified in the Main section.</td><td>Yes (Conditional)

Note: This field is required when you want to create an audit event whenever the specified element is updated.</td></tr>
</table>

<table>
<tr><th>Field Name</th><th>Field Description</th><th>Mandatory (Yes or No)</th></tr>
<tr><td>Element Value</td><td>Used to specify the element value depending on the element type. The following table lists the element value that you must specify depending on the element type:

<table><tr><th>Element Type</th><th>Element Value</th></tr><tr><td>Business Object Status</td><td>Used to indicate the status of the entity business object which you want to monitor for auditing. The system then creates an audit event whenever the entity is transitioned to the specified status.</td></tr><tr><td>Characteristic</td><td>Used to indicate the characteristic type which you want to monitor for auditing. Here, you must specify a characteristic type where the characteristic entity is set to the respective entity. The system then creates an audit event whenever the specified characteristic is defined, edited, or removed from the entity.

Note: The Search ( ) icon appears corresponding to the Element Value field. On clicking the Search icon, the Characteristic Type Search window appears.</td></tr></table>
Note: This field does not appear when the Field option is selected from the Element Type list.</td><td>Yes (Conditional)

Note: This field is required when you want to create an audit event whenever the specified element is updated.</td></tr>
</table>

Note: The **Audit Elements for Entity Update** section is enabled only when the **Update All** option is not selected in the **Main** section.

2. Enter the required details in the **Audit Elements for Entity Update** section.
3. If you want to add more than one element for auditing in the audit event type, click the **Add** ( ) icon and then repeat step 2.

Note: However, if you want to remove an element from the audit event type, click the **Delete** ( ) icon corresponding to the element.

Related Topics

For more information on...	See...
How to define an audit event type	*Defining an Audit Event Type* on page 91

For more information on...	See...
How to edit an audit event type	*Editing an Audit Event Type* on page 99
How to copy an audit event type	*Copying an Audit Event Type* on page 101

Defining a Characteristic for an Audit Event Type

Prerequisites

To define a characteristic for an audit event type, you should have:

- Characteristic types defined in the application (where the characteristic entity is set to **Audit Event Type**)

Procedure

To define a characteristic for an audit event type:

1. Ensure that the **Characteristics** section is expanded when you are defining, editing, *o*r copying an audit event type.

   The **Characteristics** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types where the characteristic entity is set to **Audit Event Type**.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the audit event type.
Characteristic Value	Used to specify the value for the characteristic type. **Note:** If you select a predefined characteristic type, the **Search** () icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, the **Predefined Characteristic Search** window appears where you can search for a predefined characteristic value. On specifying the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the audit event type.

2. Enter the required details in the **Characteristics** section.
3. If you want to define more than one characteristic for the audit event type, click the **Add** () icon, and then repeat step 2.

   Note: However, if you want to remove a characteristic from the audit event type, click the **Delete** () icon corresponding to the characteristic.

Related Topics

For more information on...	See...
How to define an audit event type	*Defining an Audit Event Type* on page 91

For more information on...	See...
How to edit an audit event type	*Editing an Audit Event Type* on page 99
How to copy an audit event type	*Copying an Audit Event Type* on page 101

Editing an Audit Event Type

You can edit an audit event type only when an audit event is not yet created using the audit event type. However, if an audit event is created using the audit event type, you can only do the following:

- Change the status from **Active** to **Inactive** and vice-versa
- Attach algorithms (if required) to the audit event type
- Define, edit, or remove characteristics from the audit event type

Prerequisites

To edit an audit event type, you should have:

- Values defined for the **AUDIT_USAGE_FLG** lookup field

Procedure

To edit an audit event type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Audit Event Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Audit Event Type** sub-menu.

   The **Audit Event Type** screen appears.

4. In the **Audit Event Type List** zone, click the **Edit** ( ) icon in the **Edit** column corresponding to the audit event type whose details you want to edit.

   The **Audit Event Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic information about the audit event type.
   - **Algorithms** - Used to associate algorithms with the audit event type.
   - **Audit Elements for Entity Update** - Used to list the elements from the entity business object which you want to monitor for auditing while updating the entity.

     Note: The **Audit Elements for Entity Update** section is enabled only when the **Update All** option is not selected in the **Main** section.

   - **Characteristics** - Used to define characteristics for the audit event type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Audit Event Type	Displays the audit event type.	Not applicable
Business Object	Indicates the business object which is used while defining the audit event type.	Not applicable
Description	Used to specify the description for the audit event type.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Detailed Description	Used to specify additional information about the audit event type.	No
Status	Used to indicate the status of the audit event type. The valid values are: • Active • Inactive **Note:** You can create only one audit event type in the **Active** status for an entity business object.	Yes
Entity Business Object	Indicates the business object for which the audit event type is defined.	Not applicable
Audit Usage	Used to indicate the purpose of auditing the entity business object. The valid value is: • Premium Calculation	Yes
Update All	Used to indicate whether you want to create an audit event whenever any field, characteristic, or status of the entity is updated.	No
Add Action	Used to indicate whether you want to create an audit event whenever the entity is added.	Yes (Conditional) **Note:** This field is required when you want to create an audit event while adding the entity.
Delete Action	Used to indicate whether you want to create an audit event whenever the entity is deleted.	Yes (Conditional) **Note:** This field is required when you want to create an audit event while deleting the entity.

Tip: Alternatively, you can access this screen by clicking the **Edit** button in the **Audit Event Type** zone.

5. Modify the required details in the **Main** section.
6. Associate or disassociate the required algorithms from the audit event type.

Note: If an audit event is created using the audit event type, you cannot edit or remove algorithms from the audit event type. In such case, you can only associate additional algorithms with the audit event type.

7. Add, edit, or remove elements from the audit event type, if required.

Note:

The **Audit Elements for Entity Update** section is enabled only when the **Update All** option is not selected in the **Main** section.

You cannot add, edit, or remove elements from the **Audit Elements for Entity Update** section when an audit event is already created using the audit event type.

8. Define, edit, or remove characteristics from the audit event type, if required.

> **Note:** You cannot define, edit, or remove characteristics from the **Characteristics** section when an audit event is already created using the audit event type.

9. Click **Save**.

   The changes made to the audit event type are saved.

Related Topics

For more information on...	See...
Audit Event Type screen	*Audit Event Type* on page 86
Audit Event Type List zone	*Audit Event Type List* on page 88
Audit Event Type zone	*Audit Event Type* on page 89
How to associate an algorithm with an audit event type	*Associating an Algorithm with an Audit Event Type* on page 93
How to add an element for auditing in an audit event type	*Adding an Element for Auditing in an Audit Event Type* on page 94
How to define a characteristic for an audit event type	*Defining a Characteristic for an Audit Event Type* on page 98

Copying an Audit Event Type

Instead of creating an audit event type from scratch, you can create a new audit event type using an existing audit event type. This is possible through copying an audit event type. On copying an audit event type, the details including the algorithms, audit elements, and characteristics are copied to the new audit event type. You can then edit the details, if required.

Prerequisites

To copy an audit event type, you should have:

- Audit event type (whose copy you want to create) defined in the application
- Entity business objects defined in the application
- The **Eligible for Audit Event** option type of the entity business object (for which you want to define an audit event type) set to **Y**

  > **Note:** The **Eligible for Audit Event** option type is available for the entity business object only when the **Valid BO Option Type** option type of the respective maintenance object is set to **C1AT**.

- Values defined for the **AUDIT_USAGE_FLG** lookup field

Procedure

To copy an audit event type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Audit Event Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Audit Event Type** sub-menu.

   The **Audit Event Type** screen appears.

4. In the **Audit Event Type List** zone, click the **Duplicate** () icon in the **Duplicate** column corresponding to the audit event type whose copy you want to create.

   The **Audit Event Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic information about the audit event type.
   - **Algorithms** - Used to associate algorithms with the audit event type.
   - **Audit Elements for Entity Update** - Used to list the elements from the entity business object which you want to monitor for auditing while updating the entity.

     Note: The **Audit Elements for Entity Update** section is enabled only when the **Update All** option is not selected in the **Main** section.

   - **Characteristics** - Used to define characteristics for the audit event type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Audit Event Type	Used to specify the audit event type. **Note:** You cannot specify any special characters except hyphen and underscore in the audit event type.	Yes
Business Object	Indicates the business object which is used while defining the audit event type.	Not applicable
Description	Used to specify the description for the audit event type.	Yes
Detailed Description	Used to specify additional information about the audit event type.	No
Status	Used to indicate the status of the audit event type. The valid values are: • Active • Inactive **Note:** You can create only one audit event type in the **Active** status for an entity business object.	Yes
Entity Business Object	Used to indicate the business object for which you want to define the audit event type. **Note:** The list includes only those business objects where the **Eligible for Audit Event** option type is set to **Y**.	Yes
Audit Usage	Used to indicate the purpose of auditing the entity business object. The valid value is: • Premium Calculation	Yes
Update All	Used to indicate whether you want to create an audit event whenever any field, characteristic, or status of the entity is updated.	No

Field Name	Field Description	Mandatory (Yes or No)
Add Action	Used to indicate whether you want to create an audit event whenever the entity is added.	Yes (Conditional) **Note:** This field is required when you want to create an audit event while adding the entity.
Delete Action	Used to indicate whether you want to create an audit event whenever the entity is deleted.	Yes (Conditional) **Note:** This field is required when you want to create an audit event while deleting the entity.

Tip: Alternatively, you can copy an audit event type by clicking the **Duplicate** button in the **Audit Event Type** zone.

5. Enter the required details in the **Main** section.
6. Associate or disassociate the required algorithms from the audit event type.
7. Add, edit, or remove elements from the audit event type, if required.

   Note: The **Audit Elements for Entity Update** section is enabled only when the **Update All** option is not selected in the **Main** section.
8. Define, edit, or remove characteristics from the audit event type, if required.
9. Click **Save**.

   The new audit event type is defined.

Related Topics

For more information on...	See...
Audit Event Type screen	*Audit Event Type* on page 86
Audit Event Type List zone	*Audit Event Type List* on page 88
Audit Event Type zone	*Audit Event Type* on page 89
How to associate an algorithm with an audit event type	*Associating an Algorithm with an Audit Event Type* on page 93
How to add an element for auditing in an audit event type	*Adding an Element for Auditing in an Audit Event Type* on page 94
How to define a characteristic for an audit event type	*Defining a Characteristic for an Audit Event Type* on page 98

Deleting an Audit Event Type

Procedure

To delete an audit event type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.
2. From the **Admin** menu, select **A** and then click **Audit Event Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Audit Event Type** sub-menu.

   The **Audit Event Type** screen appears.

4. In the **Audit Event Type List** zone, click the **Delete** ( ) icon in the **Delete** column corresponding to the audit event type that you want to delete.

   A message appears confirming whether you want to delete the audit event type.

   Note: You can delete an audit event type only when an audit event is not yet created using the audit event type.

   Tip: Alternatively, you can delete an audit event type by clicking the **Delete** button in the **Audit Event Type** zone.

5. Click **OK**.

   The audit event type is deleted.

Related Topics

For more information on...	See...
Audit Event Type screen	*Audit Event Type* on page 86
Audit Event Type List zone	*Audit Event Type List* on page 88
Audit Event Type zone	*Audit Event Type* on page 89

Viewing the Audit Event Type Details

Procedure

To view the details of an audit event type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Audit Event Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Audit Event Type** sub-menu.

   The **Audit Event Type** screen appears.

4. In the **Audit Event Type List** zone, click the **Broadcast** ( ) icon corresponding to the audit event type whose details you want to view.

   The **Audit Event Type** zone appears.

5. View the details of the audit event type in the **Audit Event Type** zone.

Related Topics

For more information on...	See...
Audit Event Type screen	*Audit Event Type* on page 86
Audit Event Type List zone	*Audit Event Type List* on page 88
Audit Event Type zone	*Audit Event Type* on page 89

Audit Event (Used for Searching)

The **Audit Event** screen allows you to search for an audit event using various search criteria. It contains the following zone:

- *Search Audit Event* on page 105

Through this screen, you can navigate to the following screen:

- *Audit Event (Used for Viewing)* on page 108

Related Topics

For more information on...	See...
How to search for an audit event	*Searching for an Audit Event* on page 107
How to view the details of an audit event	*Viewing the Audit Event Details* on page 108

Search Audit Event

The **Search Audit Event** zone allows you to search for an audit event using various search criteria. It contains the following two sections:

- **Search Criteria** - The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Search By	Used to indicate whether you want to search for an audit event using the audit event or entity details. The valid values are: • Audit Event Details • Entity Details **Note:** By default, the **Audit Event Details** option is selected.	Yes
Audit Event ID	Used to search for a particular audit event. **Note:** This field appears only when the **Audit Event Details** option is selected from the **Search By** list.	No
Created From	Used to search audit events which are created from a particular date onwards.	No
Status	Used to search audit events with a particular status. The valid values are: • Canceled • Complete • Error • Pending **Note:** This field appears only when the **Audit Event Details** option is selected from the **Search By** list.	No

Field Name	Field Description	Mandatory (Yes or No)
Created To	Used to search audit events which are created till a particular date.	No
Audit Action	Used to search audit events which are created while adding, updating, or deleting an entity. The valid values are: • Add Row • Change Row • Delete Row **Note:** This field appears only when the **Audit Event Details** option is selected from the **Search By** list.	No
Entity Business Object	Used to search audit events which are created for a particular entity business object. **Note:** The list includes only those business objects where the **Eligible for Audit Event** option type is set to **Y**. This field appears only when the **Entity Details** option is selected from the **Search By** list.	No

Note: You must specify at least one search criterion while searching for an audit event.

- **Search Results** - On clicking the **Search** button, the search results appear based on the specified search criteria. The **Search Results** section contains the following columns:

Column Name	Column Description
Creation Date Time	Displays the date and time when the audit event is created.
Audit Event Information	Displays information about the audit event. **Note:** It has a link. On clicking the link, the **Audit Event** screen appears where you can view the details of the respective audit event.
Status	Indicates the status of the audit event. The valid values are: • Canceled • Complete • Error • Pending

Column Name	Column Description
Audit Action	Indicates the action due to which the audit event is created for the entity. The valid values are: • Add Row • Change Row • Delete Row
Maintenance Object	Indicates the maintenance object of the entity business object for which the audit event is created. **Note:** This column appears only when the **Entity Details** option is selected from the **Search By** list.

Note: Pagination is used to display limited number of records in the **Search Results** section. You can use the navigation links, such as **Previous** and **Next** to navigate between pages.

Related Topics

For more information on...	See...
How to search for an audit event	*Searching for an Audit Event* on page 107
How to view the details of an audit event	*Viewing the Audit Event Details* on page 108

Searching for an Audit Event

Prerequisites

To search for an audit event, you should have:

- Entity business objects defined in the application
- The **Eligible for Audit Event** option type of the entity business object set to **Y**

Note: The **Eligible for Audit Event** option type is available for the entity business object only when the **Valid BO Option Type** option type of the respective maintenance object is set to **C1AT**.

Procedure

To search for an audit event:

1. Click the **Menu** link in the **Application** toolbar.

   A list appears.

2. From the **Main** menu, select **Financial** and then click **Audit Event**.

   The **Audit Event** screen appears.

3. Enter the search criteria in the **Search Audit Event** zone.

   Note: ORMB search engine supports wildcard search, where you can substitute the percentage (%) symbol as a stand in for any word or letter in a search criteria. You can use the '%' wildcard character in all input fields except the date and ID fields. The '%' wildcard character is suffixed automatically at the end of the partial search criteria. Therefore, you may or may not specify the wildcard character at the end of the partial search criteria. However, you have to prefix the wildcard character manually wherever required.

4. Click **Search**.

   A list of audit events that meet the search criteria appears in the **Search Results** section.

Related Topics

For more information on...	See...
Audit Event screen	*Audit Event (Used for Searching)* on page 105
Search Audit Event zone	*Search Audit Event* on page 105

Viewing the Audit Event Details

Procedure

To view the details of an audit event:

1. Search for the audit event in the **Audit Event** screen.
2. In the **Search Results** section, click the link in the **Audit Event Information** column corresponding to the audit event whose details you want to view.

   The **Audit Event** screen appears.
3. Ensure that the **Main** tab is selected.
4. View the details of the audit event in the **Audit Event** zone.

Related Topics

For more information on...	See...
How to search for an audit event	*Searching for an Audit Event* on page 107
Audit Event screen	*Audit Event (Used for Viewing)* on page 108
Audit Event zone	*Audit Event* on page 108

Audit Event (Used for Viewing)

The **Audit Event** screen allows you to view the details of an audit event. In addition, it allows you to cancel an audit event which is in the **Error** status. It contains the following tabs:

- *Audit Event - Main* on page 108
- *Audit Event - Log* on page 110

Related Topics

For more information on...	See...
How to view the details of an audit event	*Viewing the Audit Event Details* on page 108
How to cancel an audit event	*Canceling an Audit Event* on page 111

Audit Event - Main

The **Main** tab displays information about the audit event. It contains the following zone:

- *Audit Event* on page 108

Audit Event

The **Audit Event** zone displays the details of the audit event. It contains the following sections:

- **Main** - Displays basic information about the audit event. It contains the following fields:

Field Name	Field Description
Audit Event Information	Displays information about the audit event.
Audit Event Type	Indicates the audit event type using which the audit event is created. **Note:** It has a link. On clicking the link, the **Audit Event Type** screen appears where you can view the details of the respective audit event type.
Status	Indicates the status of the audit event. The valid values are: • Canceled • Complete • Error • Pending
Effective Date	Displays the effective date which is stamped corresponding to the audit event in the **C1_AUDIT_EVENT** table. **Note:** If the entity has a start date, then the effective date is set to the entity's start date. But, if the entity does not have a start date, then the effective date is set to the system date.
Audit Action	Indicates the action due to which the audit event is created for the entity. The valid values are: • Add Row • Change Row • Delete Row
Maintenance Object	Indicates the maintenance object of the entity business object for which the audit event is created. **Note:** It has a link. On clicking the link, the **Maintenance Object** screen appears where you can view the details of the respective maintenance object.

- **Characteristics** - Lists the characteristics defined for the audit event. It contains the following columns:

Column Name	Column Description
Characteristic Type	Indicates the characteristic type.
Characteristic Value	Displays the value of the characteristic type.

- **Record Actions** - This section contains the following button:

Button Name	Button Description
Cancel	Used to cancel the audit event. **Note:** The **Cancel** button appears only when the audit event is in the **Error** status.

- **Record Information** - This section contains the following fields:

Field Name	Field Description
Business Object	Indicates the business object using which the audit event is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.
Status Date/Time	Displays the date and time when the audit event status is updated.
Create Date/Time	Displays the date and time when the audit event is created.

Related Topics

For more information on...	See...
How to cancel an audit event	*Canceling an Audit Event* on page 111

Audit Event - Log

The **Log** tab contains the following zone:

- *Audit Event Log* on page 110

Audit Event Log

The **Audit Event Log** zone lists the complete trail of actions performed on the audit event. It contains the following columns:

Column Name	Column Description
Creation Date Time	Displays the date and time when the action was performed on the audit event.
Details	Displays the details about the action performed on the audit event.
User	Indicates the user who has performed the action on the audit event.
Log type	Indicates the type of the log.
Related Object	Indicates the object or entity which is created when the action is performed on the audit event. **Note:** At present, no data appears in this column. The implementation team can build the custom logic to meet the business requirements.
Status Reason	Indicates the reason why the status of the audit event is changed. **Note:** At present, no data appears in this column. The implementation team can build the custom logic to meet the business requirements.

Related Topics

For more information on...	See...
How to view the log of an audit event	*Viewing the Log of an Audit Event* on page 111

Canceling an Audit Event

There might be situations when an erroneous audit event is created in the system. In such case, the system provides you with an ability to cancel such audit events. However, note that you can cancel an audit event only when the audit event is in the **Error** status.

Prerequisites

To cancel an audit event, you should have:

- Cancel reasons defined in the application

Note: While cancelling an audit event, you need to specify the reason why you want to cancel the audit event. You can select the appropriate cancel reason only when you have defined the reasons for the **Canceled** status of the **C1-AuditEvent** business object in the **Status Reason** screen.

Procedure

To cancel an audit event:

1. Search for the audit event in the **Audit Event** screen.
2. In the **Search Results** section, click the link in the **Audit Event Information** column corresponding *t*o the audit event which you want to cancel.

   The **Audit Event** screen appears.
3. Ensure that the **Main** tab is selected.
4. Click the **Cancel** button in the **Audit Event** zone.

   The **Status Reason** window appears. It contains the following field:

Field Name	Field Description	Mandatory (Yes or No)
Status Reason	Used to indicate the reason why you want to cancel the audit event.	Yes

   Note: The **Cancel** button appears only when the audit event is in the **Error** status.
5. Select the cancel reason from the **Status Reason** list.
6. Click **Save**.

   The status of the audit event is changed to **Canceled**.

Related Topics

For more information on...	See...
How to search for an audit event	*Searching for an Audit Event* on page 107
Audit Event screen	*Audit Event (Used for Viewing)* on page 108
Audit Event zone	*Audit Event* on page 108

Viewing the Log of an Audit Event

Procedure

To view the log of an audit event:

1. Search for the audit event in the **Audit Event** screen.

2. In the **Search Results** section, click the link in the **Audit Event Information** column corresponding to the audit event whose log you want to view.

   The **Audit Event** screen appears.

3. Click the **Log** tab.

   The **Log** tab appears.

4. View the complete trail of actions performed on the audit event in the **Audit Event Log** zone.

Related Topics

For more information on...	**See...**
How to search for an audit event	*Searching for an Audit Event* on page 107
Audit Event screen	*Audit Event (Used for Viewing)* on page 108
Audit Event Log zone	*Audit Event Log* on page 110

Chapter 3

Defining Financial Transaction Options

Topics:

Bills, payments and adjustments share one very important trait - they affect how much your customers owe. This section explains the financial design of the system and describes how to set up the tables that control the financial impact of these transactions.

Note: The tables in this section are the first of many that must be set up before you can create bills and apply payments. In this section, we limit the discussion to those tables that control the financial impact of bills, payments and adjustments. In later sections, we describe the tables that control other billing-related functions like rates. It is only after all of these tables are set up that you will be able to generate the various financial transactions.

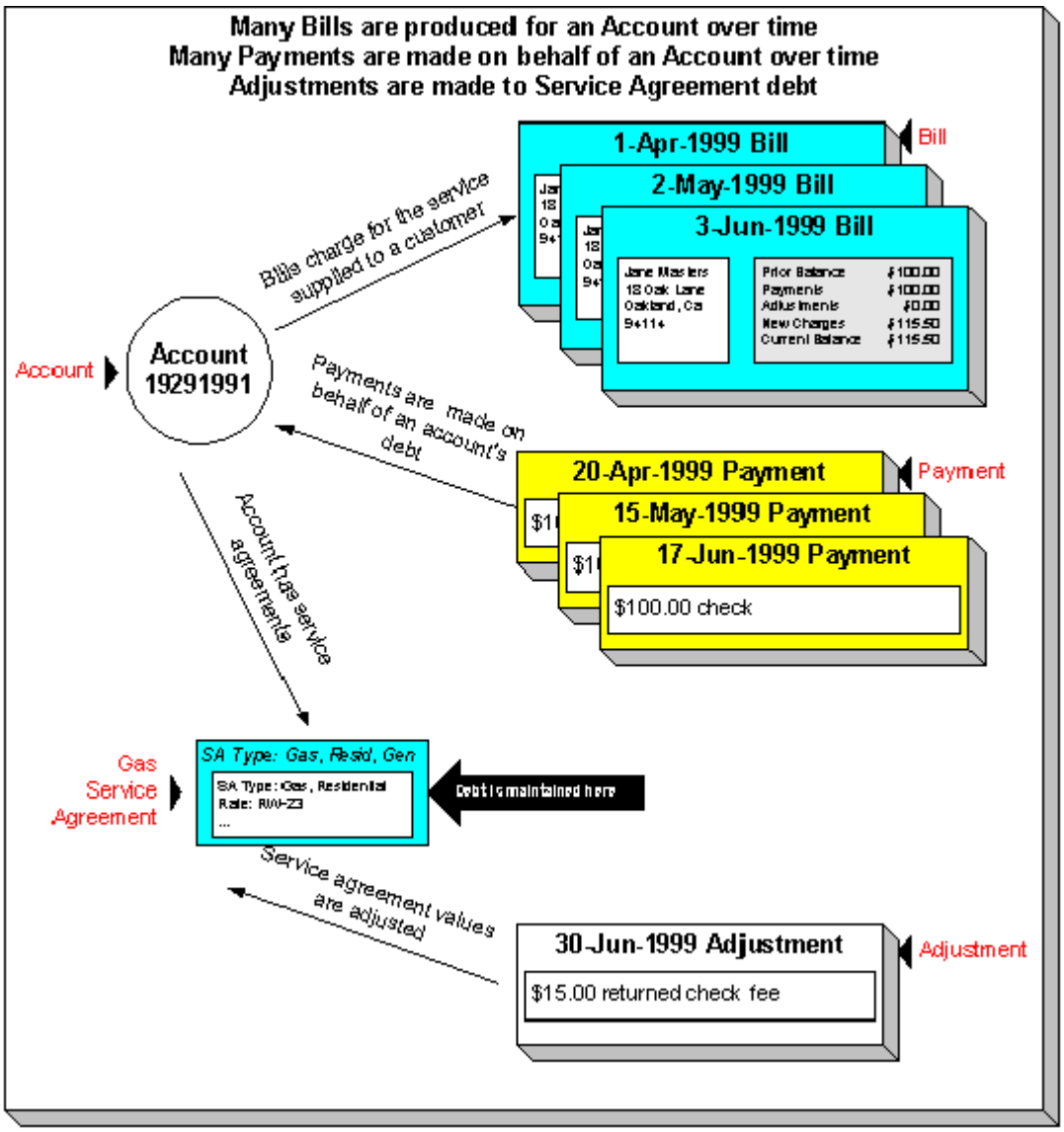
The Financial Big Picture

This section provides an overview of the relationship between an account and the various financial transactions that influence how much a customer owes.

Caution: If your organization practices cash accounting for payables (i.e., you only pay the taxing authority when you get paid), refer to *Payables Cash Accounting*. If you organization practices open-item accounting (i.e., payments must be matched to bills), refer to *Open Item Accounting*.

Bills, Payments & Adjustments

The following diagram illustrates the relationship between an account and its financial transactions:



The following concepts are illustrated above:

Bills are produced for accounts Over time, many bills may be produced for an account. For more information about a bill, see *Bill Details*.

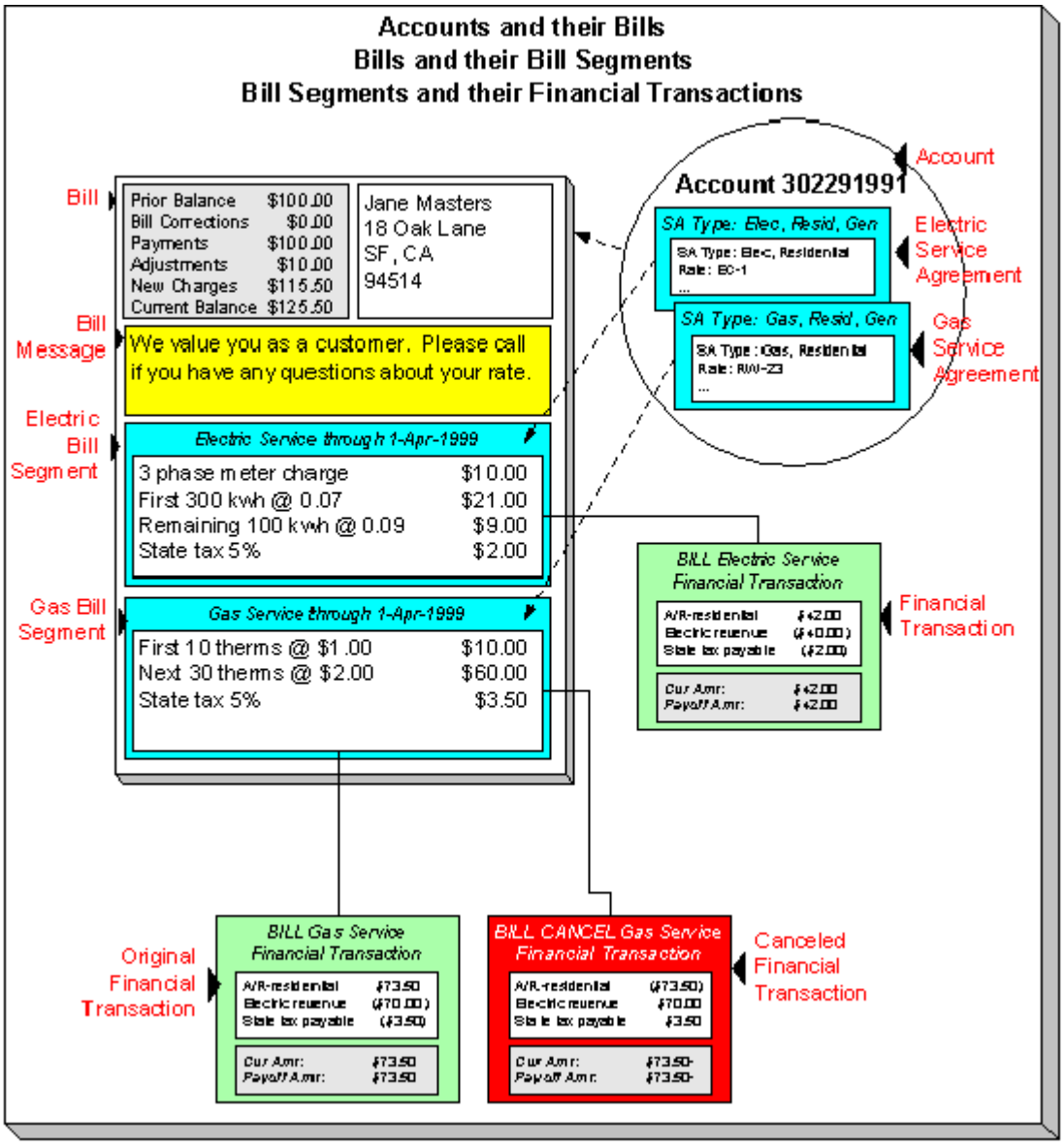
Payments are made for accounts Over time, many payments may be applied to an account's debt. For more information about a payment, see *Payment Details*.

Contracts have debt The system maintains debt on each individual contract. An account's debt is the sum of its contracts' debt.

Contracts are adjusted Over time, the debt that is stored on an account's contract(s) may be adjusted. For more information about an adjustment, see *Adjustment Details*.

Bill Details

The following diagram illustrates the relationship between an account and its bills:



The following concepts are illustrated above:

A bill is produced for an account Over time, many bills are produced for an account. A bill charges for the services supplied to a customer. The above illustration shows a single bill.

Bills contain bill segments A bill typically contains one bill segment for every active contract linked to its account. The only time this is not true is when contracts for different frequencies exist. For example, an account with a monthly and a quarterly contract will only have 4 bills a year that contain both bill segments; the other months' bills will contain a single bill segment for the monthly contract.

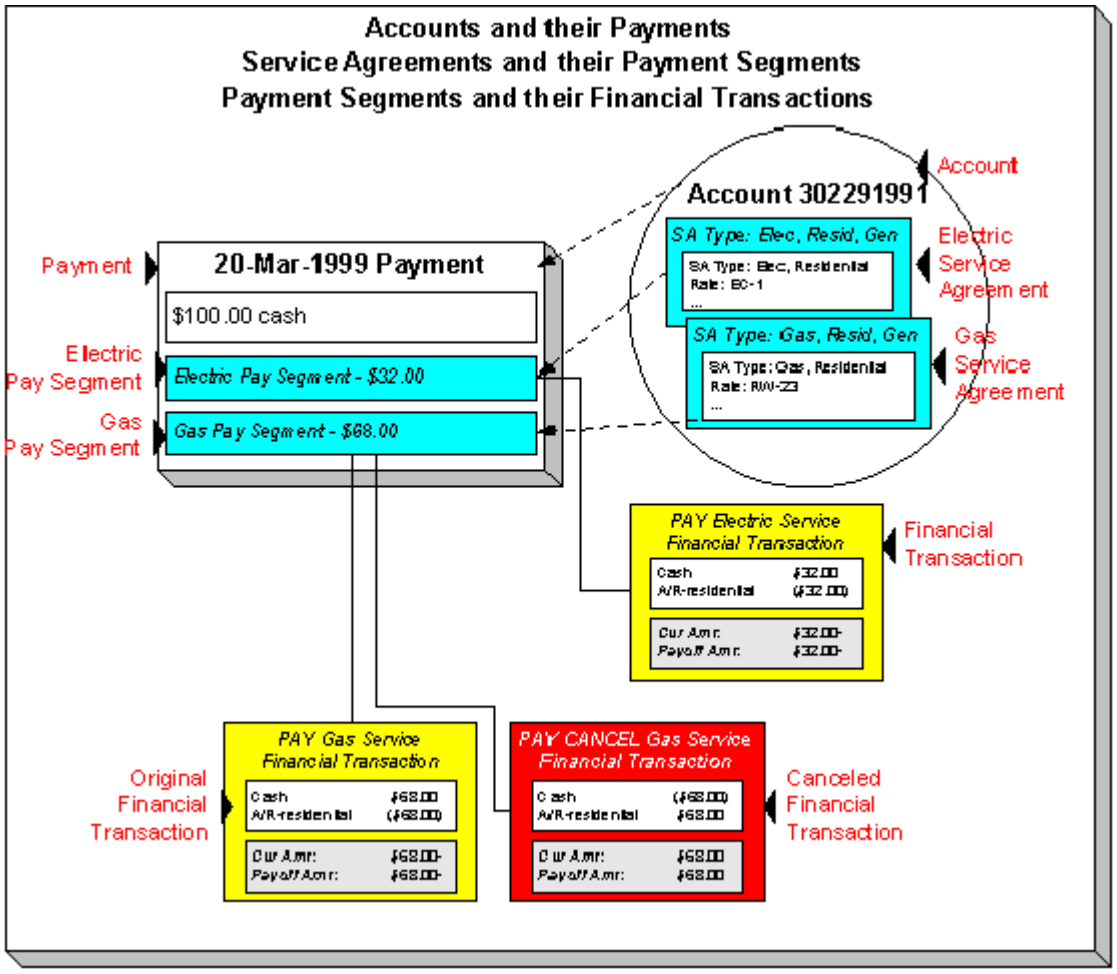
Bill segments contain calculation details A bill segment contains information showing how the segment was calculated and how it should be printed on the customer's bill.

A bill segment has a financial transaction A bill segment has a related financial transaction. A financial transaction contains the financial effects of the bill segment on the contract's current and payoff balances and on the general ledger.

Canceling a bill cancels the financial tran. If the bill segment is eventually cancelled, another financial transaction will be linked to the bill segment to reverse its original financial transaction. The cancellation financial transaction appears on the next bill produced for the account as a bill correction.

Payment Details

The following diagram illustrates the relationship between an account and its payments:



The following concepts are illustrated above:

Payments are made for accounts Over time, many payments may be applied to an account's debt. The above illustration shows a single payment.

Payments contain payment segments A payment contains one payment segment for every contract to which the payment is distributed. For a customer who pays in full, the number of payment segments will coincide with the number of bill segments on the bill being paid.

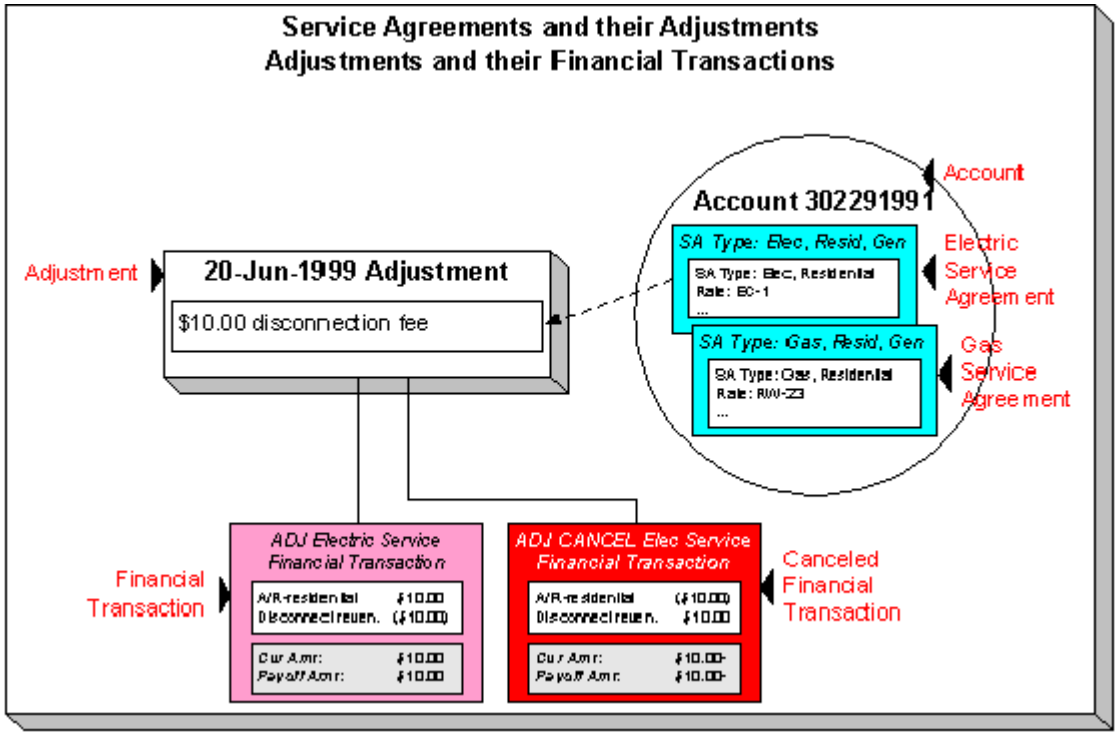
A pay. segment has a financial transaction A payment segment has a related financial transaction. A financial transaction contains the financial effects of the segment on the contract's current and payoff balances and on the general ledger.

Canceling a payment cancels the fin. tran. If the payment is eventually cancelled, another financial transaction will be linked to the related payment segment(s) to reverse their financial effect. The cancellation financial transaction appears on the next bill produced for the account as a negative payment.

Fastpath: A payment cannot be applied to an account's debt without an associated payment event. Refer to *The Big Picture of Payments* for more information.

Adjustment Details

The following diagram illustrates the relationship between an account and its adjustments:



The following concepts are illustrated above:

Contracts have adjustments Over time, a contract may have many adjustments. The above illustration shows a single adjustment to one of the account's contracts.

An adjustment has a financial transaction An adjustment has a related financial transaction. The financial transaction contains the financial effects of the adjustment on the contract's debt and on the general ledger.

Canceling an adjust. cancels the fin. tran. If the adjustment is eventually canceled, another financial transaction will be linked to the adjustment to reverse its financial effect. The cancellation financial transaction appears on the next bill produced for the account as an adjustment.

Current Amount versus Payoff Amount

A financial transaction contains two very important attributes: payoff amount and current amount. These attributes contain the grand total of how much the customer owes.

- Current amount contains how much the customer THINKS THEY OWE.
- Payoff amount contains how much the customer REALLY OWES.

You may be wondering when these two values can be different? Well, for most financial transactions, these values are the same. These values differ under the following situations:

- When a bill segment charges a customer for a charitable contribution, payoff amount will be zero because the customer doesn't really owe anything (they don't have to contribute if they don't want to). Current amount will be equal to the agreed charitable contribution amount (the customer thinks they owe the contribution).
- When a bill segment charges a customer for a deposit, payoff amount will be zero because the customer doesn't really owe anything (billed deposits are typically not viewed as being a receivable). Current amount will be equal to the amount billed (the customer thinks they owe the deposit amount).
- When a bill segment charges a customer who participates in a levelized payment program (e.g., budget billing or non-billed budgets) the two "amounts due" will contain different values. Payoff amount is equal to how much the customer really owes for the service they consumed; current amount is equal to how much they think they owe in accordance with their monthly budget.

A perhaps easier way to view these two attributes is to consider payoff amount as the "cash out amount", i.e., the amount the customer would owe the utility if they wanted to clear up all debt. The current amount contains the amount the customer thinks they owe. If you're still struggling with the difference, think about your monthly Visa bill: it contains a monthly minimum payment and the total amount owed. The minimum payment is the current amount; the total amount owed is the payoff amount.

The topics in this section provide more information about these two fields.

What Controls What Gets Booked To Current And Payoff Amount?

As described in *Bill Details*, every bill segment has a sibling financial transaction. The financial transaction defines the bill segment's affect on current and payoff amounts due. The system populates these two fields as per the Financial Transaction Algorithm defined on the bill segment's bill segment type.

Fastpath: For more information, refer to *Billing - Current Balance versus Payoff Balance* and *Designing and Defining Bill Segment Types*.

As described in *Payment Details*, every payment segment has a sibling financial transaction. The financial transaction defines the payment segment's affect on current and payoff amounts due. The system populates these two fields as per the Financial Transaction Algorithm defined on the payment segment's payment segment type.

Fastpath: For more information, refer to *Payment - Current Balance versus Payoff Balance* and *Setting Up Payment Segment Types*.

As described in *Adjustment Details*, every adjustment has a sibling financial transaction. The financial transaction defines the adjustment's affect on current and payoff amounts due. The system populates these two fields as per the Financial Transaction Algorithm defined on the adjustment's adjustment type.

Fastpath: For more information, refer to *Adjustments - Current Balance versus Payoff Balance* and *SettingUp Adjustment Types*.

Arrears

The system keeps track of the age of each customer's debt *to the day*. For example, if a customer hasn't paid their last two bills, the customer's aged debt might look as follows:

- $124.50: 22 days old
- $213.41: 51 days old

Please be aware that it is the current balance (i.e., what the customer thinks they owe) that is aged. Also keep in mind that the moment an FT is `frozen`, it impacts a customer's current balance.

The system represents aged debt in a variety of ways of the various transactions in the system. On the *Current Context Zone* arrears are shown in a colorful bar (where each color corresponds to different aged buckets):

Whereas on *Contract - Main Information*, aged debt is shown in a grid:

Days Old	Arrears Amount
New Charge	$102.38
30 Future	$102.38
1	$5,407.94
+150	$166.38

The grid method is used on many pages throughout the system. The following rows may appear in the grid:

- A row labeled `New Charge` highlights all debt that hasn't started aging yet. For example, if you've created a late payment charge and it hasn't appeared on one of the customer's bills, it will be classified as a New charge until the next bill is completed for the customer (unless a user overrides the late payment charge's arrears date by drilling into the financial transaction).
- A row with a label containing `n Future` (where n is the number of days) appears if there is "future debt". Future debt is very rare and can only exist if a debit financial transaction has a future arrears date. Financial transactions can receive a future arrears date if a bill is completed with a future date or if a user overrides a financial transaction's arrears date with a future date).
- A row that contains a number (and nothing else) represents debt that has started aging. The number is the age of the respective debt. In the above example, the customer has 1 day old debt, and debt that is more than 150 days old. Notice that the 150 day old debt is prefixed with a `+`. This means that the related debt is more than 150 days old. This age limit is controlled by a field on *Installation Options - CC* called "Oldest Bucket Age". This field limits the number of days the system will age debt. For example, if you set this field to 150, the system will never age an FT more than 150 days (and all debt that's older than 150 days will be classified as 150 day old debt). Also note, the aged debt bar that appears on *Current Context Zone* only ages debt a maximum of 60 days.
- A row with a label of `Disputed` appears if the account is an *open-item* customer and this customer has *disputed* financial transactions.

Fastpath: Refer to *Financial Transactions And Aged Debt* for more information.

GL Accounting Information

Be aware that if payoff amount is non-zero, the financial transaction has general ledger detail lines.

There are unusual financial transactions whose payoff amount is zero, but still affect the general ledger:

- Bill segments for company usage do not impact payoff amount (because your organization doesn't really owe itself anything). However, the GL is affected.
- Payment segments for charitable contributions (created when your customers contribute extra money to a charity) do not impact payoff amount. Why? Because payoff amount is never debited when a charitable contribution is billed (the customer doesn't truly owe you for this receivable). It's only when the customer pays the contribution that the GL is impacted (debit cash, credit charitable contribution payable).
- If the contract has a special role of `Loan`, the financial transaction algorithms supplied with the base package transfer the current amount between the long-term receivables and the short-term receivables in the GL. This allows the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). Refer to *Payoff Balance and Current Balance for Loans* for more information.

The effect on your GL is controlled by the financial transaction algorithm defined on your bill segment and payment segment types.

Fastpath: Refer to *The GL Interface* for how GL account information is interfaced to the general ledger.

A Complicated Example

The financial ramifications of a revolving charge account are predictable (if you're an accountant). The following table outlines the different financial events and their impact on the general ledger, arrearage history, and the amounts due (both current and payoff).

Event	GL Accounting	Arrearage Rule	Effect On Payoff Amt	Effect On Current Amt	Payoff Balance	Current Balance
Merchandise purchased	A/R 1000 Revenue <1000>	n/a (current amount is zero)	+1000	0	1000	0
Monthly bill	A/R 10 Int. Rev <10>	$120 aged accordingly	+10	+120	1010	120
Payment received	Cash 120 A/R <120>	$120 relieved accordingly	-120	-120	890	0

The following points describe the events in the above table:

- **Merchandise purchased.** When a customer purchases an air conditioner:
  - The system generates an adjustment to book the purchase.
  - The customer doesn't really think they owe the entire $1,000 (because they've purchased it on credit), therefore no moneys are booked to current amount. However, if the customer wanted to cash out, they would owe your organization $1,000, therefore the entire amount of the purchase is booked to payoff amount.
  - Because no money was booked to current amount, this event has no impact on the arrearage history.
- **Customer billed.** Monthly, the system calculates how much the client owes. In this example, interest is calculated to be $10 and the minimum monthly payment is set at $120.
  - The interest is posted to the GL, but principal isn't since it was booked when the merchandise was purchased.

    - The customer really thinks they owe the minimum payment amount, $120. Therefore, current amount is affected. However, if the customer were to cash out, they would owe your organization $1,000 + $10 (the interest); therefore payoff amount is affected by only $10.
    - Because current amount changed by $120, arrearage history is affected accordingly.
- **Payment received.** With any luck, the client will pay the $120 that was billed (note, they could obviously pay more).
    - The payment has a normal affect on the GL (debit cash, credit A/R).
    - The amount the customer thinks they owe decreases by $120, therefore current amount is affected by the payment amount. And, if the customer was to cash out, they would owe $120 less, therefore payoff amount is affected by the payment amount.
    - Because current amount changed by $120, arrearage history is affected accordingly.

Financial Transactions Created Between Bills

The following diagram illustrates how `frozen` financial transactions (FT's) accumulate between bills and are swept onto the next bill produced for the account (when the bill is `completed`). This example assumes

When any type of financial transaction is `frozen`, it impacts the related contract's *current and payoff balances*. If you do not want adjustments and bill segments to affect the customer's balance until they appear on the customer's next bill, refer to *Preventing Contract Balances And The GL From Being Impacted Until Bill Completion*.

Notice the balances in the financial summary of the above bill:

- The ***Prior Balance*** is the ending balance from the customer's prior bill.
- The ***Bill Corrections*** portion is blank. It contains a value if you cancel / rebill a bill segment that appeared on an earlier bill.
- The ***Payments*** portion shows payment financial transactions (both new payments and cancellations) that have been created since the last bill.
- The ***Adjustments*** portion shows adjustment financial transactions (both new adjustments and cancellations) that have been created since the last bill.

- The ***New Charges*** portion shows bill financial transactions that were created when the bill was created.
- The ***Current Balance*** is the total amount owed.

Fastpath: If you practice *Open Item Accounting*, refer to *Open Item Versus Balance Forward Accounting* for more information about financial transactions and bills.

Financial Transactions And Aged Debt

The system keeps track of how old a contract's current balance is in order to determine if the customer is in arrears (and therefore credit and collections processing should start).

A financial transaction (FT) impacts the related contract's current and payoff balances the moment it is `frozen`. However, some types of frozen FTs have no impact on a customer's aged debt until the next bill is completed for the account associated with the contract.

As described in the previous section, a `frozen` financial transaction (FT) waits in limbo until the customer's next bill is produced. This limbo period could be several weeks if the customer is billed infrequently. When the customer's next bill is completed, all such `frozen` FT's are linked to the bill. It is important to stress the following in respect of these FT's:

- If the FT decreases the amount of debt, the customer's aged debt is affected immediately regardless of whether the FT appears on a bill.
- If the FT increases the amount of debt, the amount the customer owes from an *aged debt* perspective may or may not be affected by the FT. There is a switch on an FT called `New Charge` that controls the arrears behavior.
    - If this switch is on, the amount of debt will be reflected as a "new charge" when you look at the customer's aged debt. This amount will remain classified as a "new charge" until the FT is swept onto a bill. The moment the FT is swept onto the customer's bill, the debt starts aging. This logic exists because you probably don't want to start aging an FT until the customer has actually seen it.
    - If this switch is off, the date on which the FT starts aging must be defined in the Arrears Date field. The Arrears Date is used to compute how many days old the debt is.

Note: Aged debt limitations. It's important to be aware that there's a field on *Installation Options - CC* called "Oldest Bucket Age" that limits the number of days old the system will age debt. For example, if you set this field to 360, the system will never age an FT more than 360 days (and all debt that's older than 360 days will be classified as 360 day old debt).

Fastpath: If you practice *Open Item Accounting*, refer to *Open Item Versus Balance Forward Accounting* for information about how open-item FT's affect aged debt.

Preventing Contract Balances And The GL From Being Impacted Until Bill Completion

It's important to understand that when any type of financial transaction is `frozen`, the related contract's balance is affected. For example:

- When a payment is `frozen`, the customer's balance is reduced.
- When an adjustment is `frozen`, the customer's balance is impacted.
- When a bill segment is `frozen`, the customer's balance is increased (typically).

For payments, there is no issue. However, for bill segments and certain types of adjustments, you may NOT want the customer's balance to be impacted until the next bill is completed. Consider the following scenarios:

- Late payment charges:

    - You can setup the system to create a late payment charge (i.e., an adjustment) say 5 days after an unpaid bill is due.
    - If the related adjustment is frozen, the customer's balance will be impacted. However, its impact will not affect *aged debt* until the next bill is completed. In other words, the amount of the frozen adjustment segment will appear as a "New Charge" until the bill is completed.
- Batch billing:
    - If a customer has multiple contracts, it's possible for one of the contracts to have a bill segment that's in `error` and the other contract's bill segment to be error-free.
    - If this happens and you have setup the bill cycle schedule to freeze bill segments if they're error-free, then you could have one bill segment `frozen` and another in `error`.
    - The `frozen` bill segment will impact the customer's balance. However, its impact will not affect *aged debt* until the bill is completed (and a bill cannot be completed until all of its bill segments are error-free). In other words, the amount of the `frozen` bill segment will appear as a "New Charge" until the bill is completed.

Caution: Aged debt is not impacted until the next bill is produced. We'd like to stress that while a frozen financial transaction impacts a customer's balance the moment it is frozen, the amount of the financial transaction appears as a "New Charge" when viewing a customer's *aged-debt*. This amount will remain classified as a "New Charge" until the next bill is completed (i.e., the customer's debt doesn't start aging until the next bill is sent to the customer).

While *aged-debt* isn't impacted by frozen FT's, the general ledger is. This is because a financial transaction is marked for *interface* to the general ledger when it is frozen. This can be problematic if you have a long period between FT freeze and bill completion (you could impact the general ledger but not impact the customer's balance). If this is unacceptable, you can setup the system to not allow certain types of FT's to be frozen until the next bill is completed. This means that neither the customer's balance nor the general ledger will be impacted until bill completion time. To do this:

- Choose the `Freeze At Bill Completion` option on *Installation Options - Billing*.
- Examine each of your *adjustment types*. Select `Freeze At Bill Completion` for those that should not impact the customer's balance or the general ledger until the next bill is completed. Select `Freeze At Will` for those that should impact the customer's balance and the GL when they are frozen. Typically, the only adjustment types for which you'd choose `Freeze At Will` option are those that cause a customer's balance to be reduced, those that are used to refund money to a customer, and those that are created at bill completion. Adjustment types for adjustments created during bill completion (e.g., by a bill completion algorithm) must have their adjustment freeze option set to `Freeze At Will`. Otherwise (i.e., if the option is `Freeze At Bill Completion`) they will not be frozen until a subsequent bill is completed.

Please be aware of the following in respect of the `Freeze At Bill Completion` options:

- If you turn on `Freeze At Bill Completion` on *Installation Options - Billing*:
    - Users will not be allowed to freeze bill segments online. This means that the freeze button will be disabled on *Bill - Main*, *Bill - Bill Segments* and *Bill Segment - Main*.
    - The `Billing` background process will not freeze bill segments until all segments on a bill are error free (and permission has been granted on the bill cycle schedule to complete bills).
    - Bill segments will exist in the `freezable` state until the bill is `completed`.
- If you turn on `Freeze At Bill Completion` on *Adjustment Type - Main*:
    - Users will not be allowed to freeze adjustments of this type online. This means that the freeze button will be disabled on *Adjustment - Main*.

- Background processes that create adjustments will not freeze this type of adjustment. Rather, the adjustments will be frozen when the next bill is completed.
- Adjustments of this type will therefore exist in the `freezable` state until the next bill is `completed`.

Note: Alerts highlight freezable FT's. Please be aware that messages appear in the *Dashboard - Financial Information Zone* to highlight the existence of freezable financial transactions.

Please be aware of the following in respect of the `Freeze At Will` options:

- If you turn on `Freeze At Will` on *Installation Options - Billing*:
  - Users will be allowed to freeze bill segments online. This means that the freeze button will be enabled on *Bill - Main*, *Bill - Bill Segments* and *Bill Segment - Main*.
  - The `Billing` background process will freeze bill segments when the individual segment is error-free (and permission has been granted on the bill cycle schedule to freeze bill segments).
  - Bill segments will exist in the `frozen` state regardless of whether the bill is completed.
  - The `frozen` bill segment's FT will be interfaced to the GL when the interface next runs.
  - All adjustment types must be also be set to `Freeze At Will` (otherwise they wouldn't get frozen).
- If you turn on `Freeze At Will` on *Adjustment Type - Main*:
  - Users will be allowed to freeze adjustments of this type online. This means that the freeze button will be enabled on *Adjustment - Main*.
  - Background processes that create adjustments will freeze this type of adjustment.
  - Adjustments of this type will exist in the `frozen` state prior to bill completion.
  - The `frozen` adjustment's FT will be interfaced to the GL when the interface next runs.

Forcing The Freeze Date To Be Used As The Accounting Date

Every financial transaction references an accounting date. The accounting date controls the accounting period to which the financial transaction is booked as described below:

- Every financial transaction references an accounting date and a contract
- Every contract references a contract type
- Every contract type references a GL division
- Every GL division references an *accounting calendar*
- The accounting calendar contains the cross reference between the accounting date specified on the financial transaction and the related accounting period in your general ledger

The accounting date is populated on financial transactions when they are initially generated. The following points describe the source of the accounting date:

- The user who creates or cancels a bill segment online defines the accounting date as part of the generation / cancel dialog (note, the current date defaults).
- Bill segments that are produced by the *BILLING* background process have their accounting date defined on the *bill cycle schedule* that caused the bill to be created.
- The user who creates or cancels an adjustment online defines the accounting date as part of the generation / cancel dialog (note, the current date defaults).
- Payments are unusual in that their financial transaction is only created when they are frozen (rather than when the payment is first distributed amongst the account's contracts). At payment freeze time, the accounting date is set to the current date.

For payments, there is no issue because the accounting date is only populated on the financial transaction when a payment is frozen. However, for bill segments and adjustments, your business practice may dictate that the freeze date should be used as the accounting date rather than the original accounting date. Alternatively, your business practice may dictate that the accounting date that's originally stamped on bill segments / adjustments should be used (unless this associated period is closed at freeze time). It's really a question of the interpretation of the local accounting rules. After you've decided on your approach, populate the **Accounting Date Freeze Option** on *Installation Options - Billing* with one of the following values:

- Choose `Always change` if the accounting date on your financial transactions should be populated with the freeze date (i.e., the current date when the financial transaction is frozen).
- Choose `Change if period is closed` if the accounting date defined when the financial transaction is generated should be used (unless the associated accounting period is closed).

Please be aware of the following in respect of your choice:

- If you choose `Always change`:
  - When a user freezes a bill segment online, they will be prompted to supply an accounting date. The current date will default, but the user can override this value.
  - When a user freezes an adjustment online, they will be prompted to supply an accounting date. The current date will default, but the user can override this value.
  - The *BILLING* background process will use the current business date as the accounting date on bill segments that it freezes.
  - Also note, if you have chosen the `Freeze At Bill Completion` **Bill Segment Freeze Option** on the *installation record*, bill segments and certain types of adjustments are frozen when a bill is completed. This means that the accounting date on the related financial transactions will be set to the completion date (because the completion date is the freeze date with this setting). Refer to *Preventing Contract Balances And The GL From Being Impacted Until Bill Completion* on page 123 for more information.
- If you choose `Change if period is closed`:
  - When a user freezes a bill segment online, they will only be prompted to supply an accounting date if the related accounting period is closed (because the accounting period closes after the bill segment is generated but before it's frozen). The current date will default, but the user can override this date.
  - When a user freezes an adjustment online, they will only be prompted to supply an accounting date if the related accounting period is closed (because the accounting period closes after the adjustment is generated but before it's frozen). The current date will default, but the user can override this date.
  - The *BILLING* background process will use the accounting date defined on the related bill cycle schedule as the accounting date on the bill segments that it creates and freezes.

How Late Payment Charges Get Calculated

Late payment charges are system-generated adjustments used to penalize a customer for late (or no) payments. This section describes how to set up the tables that control how and when late payment charges are generated. The following points describe how and when late payment charges are calculated.

- When a bill is completed, the system marks it with the date on which late payment charges will be calculated if the bill is not paid.
  - This date is calculated by adding grace days to the bill's due date. Grace days are defined on the account's *Customer Class / Division*.
  - This date will be zero if the account's *Customer Class / Division* indicates the account is not eligible for late payment charge processing.
- The late payment charge background process (referred to by the batch ID of *LATEPYMT*) selects all bills on or after their late payment charge date.

- For each such bill, the system determines if its account satisfies the late payment charge eligibility criteria defined on the account's *Customer Class / Division*. The eligibility criteria are defined in an algorithm and can therefore be as flexible as required.
- If an account is eligible for late payment charges, the system checks each of the account's contracts to determine if it is eligible for late payment charges (as defined on the contract's *Contract Type - Main Information*).
- If a contract is eligible for late payment charges, the system calls the contract type's late payment charge calculation algorithm. This algorithm should calculate the late payment charge amount, if applicable and return the calculated amount and an appropriate adjustment type to use. If this algorithm returns this information, an adjustment is generated to levy the late payment charge.

Fastpath: Refer to *Setting Up Customer Classes* for more information about how to set up an account's due days and grace period. Refer to *Contract Type - Main Information* for more information about enabling late payment charges calculations for your contracts.

Fastpath: You can update the **Late Payment Charge Details** section on the Bill - Main Information page to indicate if and when late payment charges may be levied. For more information, see *Bill - Main Information*.

Contract Type Controls Everything

The previous section illustrated three important concepts:

The true financial impact of the three financial events - bills, payments, adjustments - is at the contract level, not at the account level. This means that bills and payments are meaningless on their own. It's the contracts' bill segments, payment segments and adjustments that affect how much a customer owes.

- Every bill segment, payment segment, and adjustment has a related financial transaction. These financial transactions contain the double-sided journal entries that will be interfaced to your general ledger. They also contain the information defining how the customer's debt is affected by the financial event (i.e., current amount and payoff amount).
- A single bill can contain many bill segments, each of which may have a different frequency. For example, a bill could contain future charges, monthly retroactive charges, quarterly charges that must end on a quarter-end boundary.

You control the financial effects of the various financial events using a single field on the contract. This field is called the contract type. In this section, we describe many of the tables that must be set up before you can create a contract type.

Note: Foreshadowing. You will notice that we don't explain how to set up contract types in this section. This is because contract type controls numerous aspects of a contract's behavior in addition to its financial behavior. The non-financial aspects are discussed in later chapters. It's only after you have set up all of the control tables in this manual that you'll be able to finally define your contract types. Refer to *Setting Up Contract Types* for more information.

Caution: Take the time to define how you will record the various financial events in your general ledger before you attempt to set up these control tables. If you have simple accounting needs, this setup process will be straightforward. However, if you sell many services and use sophisticated accounting, this setup process will require careful analysis.

Setting Up Divisions

For more information, refer to the *Division* section in the **Division** folder.

Setting Up Revenue Classes

Every contract references a contract type. Amongst other things, the contract type defines a contract's revenue class. The revenue class is used when the contract's rate books revenue to different GL distribution codes based on the contract's revenue class.

Fastpath: See *Rate Component - GL Distribution* for more information about how revenue class is used to determine the GL revenue accounts referenced on a bill.

To set up revenue classes, choose **Admin Menu, Revenue Class**.

Description of Page

Enter an easily recognizable **Revenue Class ID** and **Description** for every revenue class.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_REV_CL*.

Setting Up Distribution Codes

Distribution codes simplify the process of generating accounting entries by defining valid combinations of chart of account field values.

Fastpath: Refer to *The Source Of GL Accounts On Financial Transactions* for more information about the accounting entries associated will bills, payments and adjustments.

To set up distribution codes, open **Admin Menu, Distribution Code**.

Description of Page

Enter a unique **Distribution Code** and **Description** for the distribution code.

If this distribution code is a holding account used for payables cash accounting, check the **Use For Cash Accounting** switch, and enter the actual payable **Cash Accounting Code**. The system will transfer the holding amount to this distribution code when the cash event occurs. For more information, refer to *Payables Cash Accounting*.

Define the **GL Account Algorithm** used by the system when it interfaces financial transactions that reference this distribution code to your general ledger (refer to *GLDL - Create General Ledger Download Flat File* for more information about the download process). The logic embedded in this algorithm constructs the actual GL account number. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that constructs your general ledger account number. Click *here* to see the algorithm types available for this plug-in spot.

The **Write Off Controls** control how the system writes off debt associated with the distribution code. Refer to *The Ramifications of Write Offs in the General Ledger* for an explanation of how these fields are used at write-off time.

- Define the **Division** and **Contract Type** of the contract to which bad debt associated with this distribution code should be transferred at write-off time. Note: Only contract types with a special role of `Write Off` may be selected.

- When the system transfers debt to the write-off contract defined above, the distribution code defined on this **Division / Contract Type** will be debited unless you turn on the **Override Switch**. When this switch is turned on, the system overrides the distribution code of the transfer to side of the adjustment with the distribution code associated with the debt being written off. You'd typically turn this switch on for liability distribution codes because you want to debit the original liability account when the debt is written off. Note: if this switch is on the system also overrides the characteristic type / value with the respective value associated with the debt that is being written off.

Use the **GL Account Details** scroll to define how the system constructs the GL account associated with the distribution code when it interfaces the financial transaction to your general ledger. For each distribution code, enter the following information:

- Enter the **Effective Date** of the following information.
- Define whether, on the **Effective Date**, the following information is `Active` or `Inactive`. The system will only use effective-dated information that is `Active`.
- Enter the **GL Account** that the general ledger uses to process financial transactions tagged with this distribution code.
- Enter the **Statistics Code** that should be passed to the general ledger during the GL interface for this **GL Account**.
- Select the **Validate GL Account** check box to indicate that you want to validate the GL account statically. For more information about static GL account validation, refer to *Static GL Account Validation*.
- If you have configured your installation options to indicate that *fund accounting* is `practiced`, define the **Fund** associated with this distribution code. If your installation options indicate that fund accounting is `not practiced`, the field is not visible.
- Use the grid to define characteristic values for the **Distribution Code**. To modify a characteristic, simply move to a field and change its value. The following fields display:
  - **Characteristic Type** - Indicates the type of characteristic.
  - **Characteristic Value** - Indicates the value of the characteristic.

Note: You can only choose characteristic types defined as permissible on the distribution code record. Refer to *Setting Up Characteristic Types & Their Values* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_GL_DIVISION*.

Setting Up Billable Charge Templates

A user creates a billable charge whenever a customer should be levied an ad hoc charge.

Note: Interfacing billable charges from an external system. In addition to being entered manually, billable charges can also be interfaced from an external system. You would interface billable charges if your organization provides "pass through" billing services for a service provider. Refer to *Uploading Billable Charges* for more information.

A billable charge must reference a contract. This contract behaves just like any other contract:

- **Bill segments are created for the contract.** Whenever billing is performed for an account with billable charge contracts, the system creates a bill segment for each unbilled billable charge.
- **Payments are distributed to the contract.** Payments made by an account are distributed to its billable charge contracts just like any other contract.
- **Overdue debt is monitored.** The credit and collections process monitors billable charge contracts for overdue debt and responds accordingly when overdue debt is detected.

Note: Rates can be applied to billable charges. Billable charges can be connected to a contract that also specifies a rate. The rate will be applied and lines added to the bill segment after the billable charge lines are added. For example, a rate can insert flat charges or be applied to service quantities associated with the billable charge.

Note: Taxes on top of billable charges. Rates cannot be applied to billable charge lines. If you need to perform a calculation such as applying taxes on top of the existing lines, add a service quantity (SQ) that contains the taxable amount with an SQ identifier that describes it as a taxable amount. A rate component can apply the tax to this SQ.

Fastpath: Refer to *How To Create A One-Time Invoice* for instructions describing how to create a bill for a billable charge outside of the normal bill creation process.

Billable charge *templates* exist to minimize the effort required to create a billable charge for a customer. A billable charge template contains the default bill lines, amounts and distribution codes used to levy a one-off charge.

The information on the template may be overridden by a user when the billable charge is created. For example, you can create a billable charge template to levy tree-trimming charges. This template would contain the bill lines, amounts and distribution codes associated with a tree trimming activities. Then, when you trim a tree for a customer, a user can create a billable charge using the template and override the amount to reflect the actual amount (if it differs from the norm).

Note: Templates aren't required. A billable charge can be created without a template for a truly unexpected charge.

After setting up the billable charge templates, you must indicate the contract types that can use each template. Obviously, only `billable charge` contract types (as defined on the contract type's special role) will reference billable charge templates.

Billable Charge Template - Main

Open **Admin, Billable Charge Template** to define your billable charge templates.

Description of Page

Enter a unique **Billable Charge Template** ID and **Description** for the billable charge template.

Use **Description on Bill** to define the verbiage that should print on the customer's bill above the billable charge's line item details.

Use **Currency Code** to define the currency in which the billable charge's amounts are expressed.

Use the grid to define the line item details associated with the billable charge (note, the **Total Line Amount** field is automatically calculated. It is the sum of the **Charge Amount** on each of the Line Sequence items). The following fields are required for each entry in the grid.

Sequence Line sequence controls the order in which the line items appear on the bill segment.

Description on Bill Specify the verbiage to print on the bill for the line item.

Charge Amount Specify the default amount to charge for the line item.

Show on Bill Turn this switch on if the line item should appear on the customer's printed bill. It would be very unusual for this switch to be off.

Appears in Summary Turn this switch on when the amount associated with this line also appears in a summary line.

Memo Only, No GL Turn this switch on when the amount associated with this line does not affect the GL (or the total amount owed by the customer).

Distribution Code Specify the default distribution code associated with this line item.

If you use the drill down button on the left most column in the grid, you will be taken to the Line Characteristics tab with the selected line displayed.

Fastpath: For more information about creating a billable charge, refer to *Maintaining Billable Charges*. For more information about billing billable charges, refer to *How To Create A One-Time Invoice*.

Billable Charge Template - Line Characteristics

Open **Admin, Billable Charge Template, Line Characteristics** to define your billable charge templates line characteristics.

Description of Page

The **Line Sequence** scroll defines the billable charge template line to which you wish to assign characteristic values.

To modify billable charge template line characteristics, simply move to a field and change its value. To add characteristics, press + to insert a row and then fill in the information for each field. The following fields display:

Characteristic Type The type of characteristic.

Characteristic Value The value of the characteristic.

Billable Charge Template - SQ Details

Open **Admin, Billable Charge Template, SQ Details** to define your billable charge templates service quantities.

Description of Page

To modify a template's service quantity, simply move to a field and change its value. To add a new service quantity to the billable charge template, press the + button to insert a row and fill in the information for each field. The following fields display:

Sequence Used to specify the sequence number for the UOM, Variance Parameter, and SQ Identifier combination.

UOM Used to indicate the product for which you want to create the billable charge.

Variance Parameter Used to indicate the variance parameter which you want to use along with the product for determining the price.

SQ Identifier Used to indicate the service quantity identifier that you want to use for calculating the product charges.

Service Quantity Used to specify the number of units of the service quantity.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_B_CHG_TMPLT*.

Designing and Defining Bill Segment Types

Every contract references a contract type. Amongst other things, the contract type references a bill segment type. The bill segment type controls how bill segments and their related financial transactions are created.

Caution: We strongly recommend understanding the concepts described in *The Big Picture of Billing* before setting up your bill segment types.

The topics in this section describe how to design and set up bill segment types.

What Do Bill Segment Types Do?

Bill segment types control how bill segments and their related financial transactions are created.

Designing Your Bill Segment Types

The following table contains a subset of the contract types listed under *Defining Contract Types*. However, if your reading this document from top to bottom, you probably don't know what your contract types are (they are only designed much later) and will have to forestall this task until that time.

We're going to cheat and assume you know what your contract types are and fill in the algorithms necessary to create bill segments for each contract type. After this table is complete, we will look for unique combinations of the 4 algorithms and create a bill segment type for each one.

Note: Before you can fill in the columns for your own contract types, you should be comfortable with the descriptions of the algorithms described under *Setting Up Bill Segment Types* .

Div-Contract Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm
CA/G-RES	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	*Auto cancel bad estimates*
CA/G-COM	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption
CA/G-IND	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption
CA/CABLE	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption
CA/E-COY	*Apply Rate*	*Payoff = 0 / Current = 0*	*Get Consumption From SP's*	N/A - can't estimate consumption
CA/WO-STD	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable
CA/WO-LIA	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable
CA/CHARITY	*Recurring Charge*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption
CA/PA-REGU	*Recurring Charge With Auto Stop*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption
CA/MERCH-I	*Recurring Charge With Auto Stop*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption
CA/DEP-I	*Recurring Charge For Amount To Bill*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption

Div-Contract Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm
CA/ONETIME	*Billable Charge*	*Payoff = Bill Amount / Current = Amount Due*	N/A - no consumption is needed	N/A - no consumption
CA/OVR UNDR	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable
CA/E-SUB ENR	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From Master Bill Segment*	N/A - rated sub contracts are cancelled when there master is cancelled
CA/E-SUB BC	*Billable Charge*	*Payoff = Bill Amount / Current = Amount Due*	N/A - no consumption is needed	N/A - no consumption
CA/E-FIN SETTL	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable

Now, we'll extract unique combinations of the 4 algorithms and create a bill segment type for each.

Bill Segment Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm
SP RATED	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	*Auto cancel bad estimates*
NOESTRAT	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption
COMPUSAG	*Apply Rate*	*Payoff = 0 / Current = 0*	*Get Consumption From SP's*	N/A - can't estimate consumption
BILLCHRG	*Billable Charge*	*Payoff = Bill Amount / Current = Amount Due*	N/A - no consumption is needed	N/A - no consumption
RECUR	*Recurring Charge*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption
RECUR AS	*Recurring Charge With Auto Stop*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption
RECURATB	*Recurring Charge For Amount To Bill*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption

Bill Segment Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm
SUB RATE	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From Master Bill Segment*	N/A - rated sub contracts are cancelled when there master is cancelled
SUB BC	*Billable Charge*	*Payoff = Bill Amount / Current = Amount Due*	N/A - no consumption is needed	N/A - no consumption

Just to make sure everything has been designed appropriately, we will return to our contract type samples and specify their respective bill segment types:

Div-Contract Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm	Bill Segment Type
CA/G-RES	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	*Auto cancel bad estimates*	SP-RATED
CA/G-COM	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption	NOESTRAT
CA/G-IND	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption	NOESTRAT
CA/CABLE	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From SP's*	N/A - can't estimate consumption	SP-RATED
CA/E-COY	*Apply Rate*	*Payoff = 0 / Current = 0*	*Get Consumption From SP's*	N/A - can't estimate consumption	COMPUSAG
CA/WO-STD	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	
CA/WO-LIA	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	
CA/ CHARITY	*Recurring Charge*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption	RECUR
CA/PA-REGU	*Recurring Charge With Auto Stop*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption	RECUR-AS

Div-Contract Type	Calculation Algorithm	FT Algorithm	Consumption Algorithm	Auto Cancel Algorithm	Bill Segment Type
CA/MERCH-I	*Recurring Charge With Auto Stop*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption	RECUR-AS
CA/DEP-I	*Recurring Charge For Amount To Bill*	*Payoff = 0 / Current = Bill Amount*	N/A - no consumption is needed	N/A - no consumption	RECURATB
CA/ ONETIME	*Billable Charge*	*Payoff = Bill Amount / Current = Amount Due*	N/A - no consumption is needed	N/A - no consumption	BILLCHRG
CA/OVR UNDR	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	
CA/E-SUB ENR	*Apply Rate*	*Payoff = Bill Amount / Current = Amount Due*	*Get Consumption From Master Bill Segment*	N/A - rated sub contracts are cancelled when there master is cancelled	SUB RATE
CA/E-SUB BC	*Billable Charge*	*Payoff = Bill Amount / Current = Amount Due*	N/A - no consumption is needed	N/A - no consumption	SUB BC
CA/E-FIN SETTL	N/A - non billable	N/A - non billable	N/A - non billable	N/A - non billable	

And now you're ready to set up your bill segment types.

Setting Up Bill Segment Types

To set up bill segment types, open **Admin Menu, Bill Segment Type**.

Description of Page

Enter an easily recognizable **Bill Segment Type** and **Description** for every type of bill segment.

For each bill segment type, define the **Create Algorithm**. The logic embedded in this algorithm creates the bill segment. Refer to *Designing Your Bill Segment Types* for examples.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that creates a bill segment in the appropriate manner. Click *here* to see the algorithm types available for this plug-in spot.

Caution: The **BS Create Algorithm** is a very important field as it controls how the system creates bill segments. There are some restrictions in respect of the values of certain fields on the contract type and the bill segment algorithm used on a contract type. Refer to *Require Total Amount Switch versus Bill Segment Creation Algorithm*, *Allow Recurring Charge Switch versus Bill Segment Creation Algorithm*, and *Rate Required Switch versus Bill Segment Creation Algorithm* **for more information**.

For each bill segment type, define the **Financial Algorithm**. The logic embedded in this algorithm constructs the financial transaction associated with the bill segment. Refer to *Designing Your Bill Segment Types* for examples.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that constructs the bill segment financial transaction in the appropriate manner. Click *here* to see the algorithm types available for this plug-in spot.

Fastpath: For more information about current and payoff amounts, refer to *Current Amount versus Payoff Amount*.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that retrieves consumption in the appropriate manner. Click *here* to see the algorithm types available for this plug-in spot.

The **Auto Cancel Algorithm** is used by the system when it detects that the prior bill segment contains a bad estimated read (by "bad" we mean that the current bill has a non-estimated reading that is less than the estimated end read on the prior bill segment). If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that cancels bill segments that contain poorly estimated consumption. Click *here* to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_BILL_SEG_TYP*.

Designing and Defining Deposit Classes

If you bill for deposits, you must set up one or more deposit classes. If your company does not bill for deposits, you can skip this section.

The topics in this section describe how to design and set up deposit classes.

What Do Deposit Classes Do?

A deposit class contains the business rules that govern:

- How and when deposit interest is calculated.
- How the recommended deposit amount is calculated.
- When a deposit will be automatically refunded to a customer.
- When the system will recommend a new or additional deposit.

When you link a deposit class to a contract type, you are indicating that the contract type's contracts are governed by the deposit class' business rules.

In addition to linking a deposit class to the contract types used to bill for a deposit, you must also define a deposit class on contract types whose debt is covered by a deposit. For example, if your company can apply a deposit to any type of debt, then you'd have just one deposit class - `General Deposit`. You'd link this deposit class to the deposit contract type, and to the other contract types whose debt is covered by the deposit.

Note: Non-cash deposits. You can also use the system to manage non-cash deposits (e.g., letters of credit, surety bonds, 3 rd party deposits). Non-cash deposits are held in respect of an account (and an account may have an unlimited number of non-cash deposits). Each non-cash deposit must reference a deposit class. Why? Because the system amalgamates cash and non-cash deposits when it determines if an account is holding an adequate deposit.

Designing Your Deposit Classes

A deposit class contains the business rules that govern:

- How and when deposit interest is calculated.
- How the recommended deposit amount is calculated.
- When a deposit will be automatically refunded to a customer.
- When the system will recommend a new or additional deposit.

You will need multiple deposit classes if any of the above rules / conditions differ for different types of customers. For example, if residential customers use a different recommended deposit algorithm as compared to commercial customers, you'd need one deposit class for residential and another for commercial.

You will need additional deposit classes if your customers can have multiple deposits where each deposit is restricted to a specific type of debt.

We'll design deposit classes to satisfy the needs of a theoretical company to help you understand how to design your deposit classes. The following points describe the deposit requirements of our theoretical company:

- The recommended deposit amount is 2 times the average bill (averaged over the last 12 months). This is true regardless of the type of customer or debt.
- The system should automatically refund a deposit to a customer after:
    - The deposit has been held for at least 6 months; and
    - The account's credit rating is greater than the credit rating threshold defined on the installation record (i.e., the credit rating is no longer considered bad)
- This is true regardless of the type of customer or debt.
- Interest is calculated every 6 months. The interest rate is defined using a bill factor (refer to *Setting Up Bill Factors* for more information). This is true regardless of the type of customer or debt.
- When it's time to refund a refund a deposit, all outstanding debt will be paid off first. If any moneys remain, a check should be sent to the customer for the remainder. This is true regardless of the type of customer or debt.

You may wonder why two deposit classes are needed when the rules are the same for both? Well, besides defining the applicable business rules for a deposit contract, a deposit class is defined on the contract types whose debt is covered by the deposit class' deposit. So, if you have two different types of debt where each type of debt can have its own deposit, you'd need at two deposit classes. Each deposit class would be associated with the contracts that are being secured by the deposit.

Refer to *Setting Up Deposit Classes* for a description of the various algorithms defined in respect of a deposit class.

Setting Up Deposit Classes

In the previous section, Designing Your Deposit Classes, we presented a case study that illustrated a mythical organization's deposit classes. In this section, we explain how to maintain your Deposit Classes.

Deposit Class - Main

To set up deposit classes, select **Admin Menu**, **Deposit Class**.

Description of Page

Enter an easily recognizable **Deposit Class** and **Description**.

Use **Refund Description on Bill** to define the information that appears on the bill segment produced when it's time to refund the customer's deposit.

The remaining information on this page is used by the various deposit-oriented processes.

Refer to *Deposit Class - Good Customer* for information about the **Good Customer Algorithm**.

Refer to *Deposit Class - Recommendation* for information about the **Recommendation Algorithm** and **Review Tolerance Percentage**.

Refer to *Deposit Class - Refund Method* for information about the **Refund Method Algorithm**.

Refer to *Deposit Class - Refund Criteria* for information about the **Refund Criteria Algorithm**.

Refer to *Deposit Class - Refund Interest* for information about the **Interest Refund Algorithm** and **Int Refund Interval (Months)**.

Refer to *Deposit Class - Review Method* for information about the **Review Method Algorithm**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_DEP_CL*.

Deposit Class - Good Customer

On *Deposit Class - Main* you must define the **Good Customer Algorithm** used by the system when it determines if a customer is considered good (the system recommends new / additional deposits for bad customers).

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines if a customer is considered good. Click *here* to see the algorithm types available for this plug-in spot.

Deposit Class - Recommendation

On *Deposit Class - Main* you must define the **Recommendation Algorithm** used by the system when it calculates a customer's recommended deposit amount.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that recommends deposits. Click *here* to see the algorithm types available for this plug-in spot.

The system uses the **Review Tolerance Percentage** to prevent the recommendation of small deposits by the Deposit Review background process. For example, if this field contains 10(%), the system would only recommend an additional deposit if the recommended amount was more than 10% of the existing deposit.

Deposit Class - Refund Method

On *Deposit Class - Main* you must define the **Refund Method Algorithm** used by the system when it refunds a deposit to the customer.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that refunds a deposit to a customer. Click *here* to see the algorithm types available for this plug-in spot.

Deposit Class - Refund Criteria

On *Deposit Class - Main* you must define the **Refund Criteria Algorithm** used by the system when it determines if it should automatically refund a deposit to a customer.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines if a customer qualifies for a deposit refund. Click *here* to see the algorithm types available for this plug-in spot.

Deposit Class - Refund Interest

On *Deposit Class - Main* you must define the **Interest Refund Algorithm** to define how the system calculates interest and how it refunds the interest to the customer.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that calculates interest on a deposit. Click *here* to see the algorithm types available for this plug-in spot.

Interest will be automatically calculated every X months where X is defined in **Int Refund Interval (Months)**.

Deposit Class - Review Method

On *Deposit Class - Main* you must define the **Review Method Algorithm** used by the system to determine what action to take if the system recommends a deposit (or additional deposit) amount for an account.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines the review method the system uses if it recommends a deposit or additional deposit be applied to an account. Click *here* to see the algorithm types available for this plug-in spot.

Setting Up Non-Cash Deposit Types

Non-cash deposit types are used to indicate the type of monetary instrument used for non-cash deposits.

To define your non-cash deposit types, select **Admin Menu, Non-Cash Deposit Type**.

Description of Page

To modify a non-cash deposit type, move to a field and change its value.

To add a new non-cash deposit type, insert a row, then fill in the information for each field. The following fields display:

Non-Cash Deposit Type The unique identifier of the non-cash deposit type.

Description The description of the non-cash deposit type.

Review Before Expiration This switch indicates if the system will create a ToDo entry when non-cash deposits of this type are close to expiration.

Third Party Deposit This switch indicates if the system requires a reference to a 3 rd party's deposit contract for this type of non-cash deposit.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_NCD_TYPE*.

Setting Up Payment Segment Types

Every contract references a contract type. Amongst other things, the contract type references a payment segment type. The payment segment type controls how payment segments and their related financial transactions are created. To set up payment segment types, open **Admin Menu, Payment Segment Type**.

Description of Page

Enter an easily recognizable **Payment Segment Type** and **Description** for every type of payment segment.

Fastpath: For more information about the source of the distribution codes on financial transactions, see *The Source Of GL Accounts On Financial Transactions*.

For each payment segment type, define the **Payment Segment Fin Algorithm**. The logic embedded in this algorithm constructs the actual financial transaction associated with the payment segment. Refer to *Examples of Common Payment Segment Types* for examples of how algorithms are used on common payment segment types.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that constructs the payment segment financial transaction in the appropriate manner. Click *here* to see the algorithm types available for this plug-in spot.

Fastpath: For more information about current and payoff amount, see *Current Amount versus Payoff Amount*.

Examples of Common Payment Segment Types

The following table shows several classic payment segment types used by many organizations:

Payment Segment Type	Payment Segment Financial Transaction Algorithm
Normal payment (if you practice accrual accounting). Refer to *Accrual versus Cash Accounting* for more information.	*Payoff = Pay Amount / Current = Pay Amount (no cash accounting)*
Normal payment (if you practice cash accounting). Refer to *Accrual versus Cash Accounting* for more information.	*Payoff = Pay Amount / Current = Pay Amount (plus Cash Accounting)*
Charity payment	*Payoff =0 / Current = Pay Amount (the GL is affected)*
Non-CIS Payments (When the FT is created, the distribution code and GL account to credit is retrieved from the pay). Refer to *Non CIS Payments* for more information	*Payoff = Pay Amount / Current = Pay Amount (no cash accounting)*

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_PAY_SEG_TYPE*.

Setting Up Adjustment Types

A contract's debt may be changed with an adjustment. Every adjustment must reference an adjustment type. The adjustment type contains a great deal of information that is defaulted onto the adjustment, including whether the adjustment amount is calculated. It also controls many business processes associated with the adjustment.

An adjustment type contains the business rules that govern how the adjustments must be managed by the system. Please refer to The Big Picture Of Adjustments for a complete description of how adjustment types impact the lifecycle of adjustments.

Note: When a new adjustment type is added. When you introduce a new adjustment type, you must update one or more adjustment profiles with the new adjustment type. Why? Because adjustment profiles define the adjustment types that may be levied on contracts (adjustment profiles are defined on contract types). If you don't put the adjustment type on an adjustment profile, the adjustment type can't be used on any adjustment.

Adjustment Type - Main

To set up adjustment types, open **Admin Menu, Adjustment Type**.

Description of Page

Enter a unique **Adjustment Type** ID and **Description** for the adjustment type.

The **Adjustment Amount Type** indicates whether the adjustment about is calculated or not. Select `Calculated Amount` when you want to use a rate to perform calculations to generate the adjustment amount otherwise select `Non-Calculated Amount`. Refer to *Setting Up Calculated Adjustment Types* for more information about calculated adjustments.

Enter the **Distribution Code** that references the GL account associated with the adjustment. For example, if this adjustment type is used to levy a charge for a bad check, the distribution code would reference the revenue account to which you associate such revenue. Note, the offsetting distribution code is kept on the contract type.

Note: Distribution Code for Calculated Adjustments. Depending on the algorithm used for the *calculated adjustment*, the distribution code may come from the adjustment type or the calculation lines of the algorithm. If the adjustment's calculation algorithm gets the distribution code from the calculation lines, you do not need to specify a distribution code on the adjustment type.

Fastpath: For more information about the source of the distribution codes on financial transactions, see *The Source Of GL Accounts On Financial Transactions*.

Enter the **Currency Code** for adjustments of this type.

Turn on **Sync. Current Amount** if adjustments of this type exist to force a contract's current balance to equal its payoff balance. These types of adjustments are issued before a contract's funds are transferred to a write-off contract. If this switch is on, choose an **Adjustment Fin Algorithm** that does not impact payoff balance or the GL, but does affect the contract's current balance (refer to *ADJT-CA* for an example of such an algorithm).

Enter a **Default Amount** if an amount should be *defaulted* onto adjustments of this type.

Fastpath: For more information about current and payoff amounts, refer to *Current Amount versus Payoff Amount*.

If the AP Adjustment should be recorded in respect of the customer's 1099 amounts, indicate the **A/P 1099 Flag**. This would typically be used on the adjustment used to credit the deposit contract with accrued interest. The values of this field are `Interest` and `Miscellaneous`. This type of adjustment would also have an **A/P Request Type Code** selected, as 1099 reporting is handled in A/P.

Use **Print by Default** to indicate whether you want to print the information about this type of adjustments on the account’s next bill. The **Bill ID** field appears in the **Adjustment** screen only when the **Print by Default** check box is selected on the respective adjustment type.

Use **Impact Next Bill Balance** to indicate whether the account’s next bill balance must be affected when an adjustment is created using the adjustment type.

Choose an **A/P Request Type Code** if this adjustment is interfaced to accounts payable (i.e., it's used to send a refund check to a customer). Refer to *A/P Check Request* for more information.

The **Adjustment Freeze Option** defines when adjustments can be frozen and therefore when a contract's balance and the general ledger are affected by an adjustment. Refer to *Preventing Contract Balances And The GL From Being Impacted Until Bill Completion* to understand the significance of this option. Also note, if the *installation option's* Bill Segment Freeze Option is `Freeze At Will`, this field is defaulted to `Freeze At Will` and cannot be changed.

Caution: Adjustment types for adjustments created during bill completion (e.g., by a bill completion algorithm) must have their adjustment freeze option set to `Freeze At Will`. Otherwise (i.e., if the option is `Freeze At Bill Completion`) they will not be frozen until a subsequent bill is completed.

If adjustments of this type require approval, define an **Approval Profile**. For more information, refer to *The Big Picture of Adjustment Approval* on page 153.

Enter the verbiage to appear on the printed bill in **Description on Bill**.

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all adjustments of this type. These can be used for reporting purposes or in your algorithms.

Adjustment Type - Adjustment Characteristics

To define characteristics that can be defined for adjustments of a given type, open **Admin Menu, Adjustment Type** and navigate to the **Adjustment Characteristics** tab.

Description of Page

Use the **Adjustment Characteristics** collection to define characteristics that can be defined for adjustments of a given type. Turn on the **Required** switch if the **Characteristic Type** must be defined on adjustments of a given type. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

Adjustment Type - Algorithms

To define algorithms for adjustments, open **Admin Menu, Adjustment Type** and navigate to the **Algorithms** tab.

Description of Page

The grid contains **Algorithms** that control important adjustment functions. If you haven't already done so, you must *set up the appropriate algorithms* in your system. You must define the following for each algorithm:

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

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

System Event	Optional / Required	Description
`Adjustment Cancellation`	Optional	When an adjustment is canceled an algorithm of this type may be called to do additional work. Refer to *The Lifecycle Of An Adjustment* for more information about canceling an adjustment. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Adjustment Freeze`	Optional	When an adjustment is frozen an algorithm of this type may be called to do additional work. Refer to *The Lifecycle Of An Adjustment* for more information about freezing an adjustment. Click *here* to see the algorithm types available for this system event.
`Adjustment Information`	Optional	We use the term "Adjustment information" to describe the basic information that appears throughout the system to describe an adjustment. The data that appears in "Adjustment information" is constructed using this algorithm. Plug an algorithm into this spot to override the "Adjustment information" algorithm on installation options or the system default "Adjustment information" if no such algorithm is defined on installation options. Click *here* to see the algorithm types available for this system event.
`Adj. Financial Transaction`	Required	Algorithms of this type are used to construct the actual financial transaction associated with the adjustment. The financial transaction controls the adjustment's affect on the contract's payoff and current balances. It also constructs the information that is eventually interfaced to your general ledger. Refer to *Examples of Common Adjustment Types* for examples of how algorithms are used on common adjustment types. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Default Adjustment Amount`	Optional	Algorithms of this type are used to default the adjustment amount. Refer to *Default the Adjustment Amount* for more information. Click *here* to see the algorithm types available for this system event.
`Determine Contract`	Optional	Algorithms of this type are used to find a contract for which the adjustment can be posted. This plug-in is used particularly during adjustment upload when a staging record does not identify the Contract ID. Refer to *Interfacing Adjustments From External Sources* for more information. Click *here* to see the algorithm types available for this system event.
`Resolve Suspense`	Optional	Algorithms of this type are used to automatically resolve adjustments that are in suspense. Refer to *Suspense Adjustments* for more information Click *here* to see the algorithm types available for this system event.
`Generate Adjustment`	Optional	Algorithms of this type are used to calculate the adjustment amount if an adjustment type indicates that the adjustment amount is calculated. Refer to *Setting Up Calculated Adjustment Types* for more information. Click *here* to see the algorithm types available for this system event.
`Validate Adjustment`	Optional	Algorithms of this type are used to validate information for the adjustment after it is generated. Click *here* to see the algorithm types available for this system event.

Setting Up Calculated Adjustment Types

You can use an algorithm to calculate an adjustment amount for example if you need to calculate tax on a base amount or calculate a non-sufficient funds charge based on the customer's credit rating. Because the base package algorithm calculates adjustment amounts by calling the rate application, calculated adjustments are sometimes referred to as ratable adjustments.

Note: Ratable Adjustments Appear Deceptively Simple. But, they are not. Calculated adjustments that use the base package algorithm have all the power and flexibility (and complexity) of the rate application. Anything that you can do with a rate can be applied to a calculated adjustment. For examples that illustrate the flexibility of the rate application (and therefore calculated adjustments), refer to *Rate Examples*.

Adjustment types that indicate they are calculated have a generate adjustment algorithm. The base package algorithm defines the rate schedule used to calculate the adjustment as well as any UOM, Variance Parameter or SQI parameters.

To set up calculated adjustment types using the base package generate adjustment algorithm type:

- Define the rate that performs the calculations, including the rate schedule, rate version and rate components. Refer to *How To Create A New Rate* for information.

Note: If you create your own Generate Adjustment algorithm type, you may not need to set up a rate that performs the calculations. It depends on the needs of your algorithm type.

- Create a Generate Adjustment algorithm (refer to *Setting Up Algorithms*) that references the base package algorithm type that generates calculated adjustments (see the table above).
- If you want the generation algorithm's calculation lines to provide the distribution codes when the adjustment is posted to the GL, create an Adjustment Financial Transaction algorithm (refer to *Setting Up Algorithms*) that references an algorithm type that creates the adjustment's financial transactions using the calculation lines. A parameter of the adjustment financial transaction algorithm determines whether the distribution codes are taken from the adjustment type (AT) or calculation lines (CL). The system comes supplied with several sample algorithm types that *create adjustment financial transactions*.
- Create an adjustment type where the **Adjustment Amount Type** is `Calculated Amount`, the `Generate Adjustment` event references the generation algorithm created above, and the `Adj. Financial Transaction` event references the adjustment financial transaction algorithm created above.

Examples of Common Adjustment Types

The following table shows several classic adjustment types used by many organizations:

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
NSF	NSF revenue	Your NSF fee	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	No	No

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
LPC	Late payment charge revenue	N/A	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	No	No
CONNECT	Connection charge revenue	Your connection charge	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	No	No
CUSTREL	Customer relationship expense	N/A	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	No	No
ADDCHARG	Misc Revenue	N/A	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	No	No

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
XFER	Balance transfer clearing	N/A	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	No	No
WO SYNC	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	No	No	No
REFUNDAP	A/P clearing	N/A	Payoff = Current = Adj. Amt Refer to *ADJT-NM* for an example of such an algorithm type	Yes	Yes	No
DPA FIX	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	Yes	No	No

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
CHARITFX	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	Yes	No	No
BUDG ON	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	Yes	No	No
BUDG OFF	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	Yes	No	No
BUDG FIX	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	Yes	No	No

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
DEPOSREF	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	Yes	No	No

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
DEPOSINT	Interest expense	N/A	Payoff = Current = Adj. Amt. Refer to *ADJT-NM* for an example of such an algorithm type. Or Payoff Amt = Adj Amt / Current Amt = 0. Refer to *ADJT-TA* for an example of such an algorithm type. Use the first method if you want to have the interest reflected as a credit balance on the customer's bill. Use the second method if you roll the interest amount into the customer's existing deposit on hand.	Yes	No	Yes

Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm	Print On Bill	A/P Adjust-ment	1099
DEPFIXCR	N/A	N/A	Payoff = 0 / Current = Adj. Amt Refer to *ADJT-CA* for an example of such an algorithm type	No	No	No

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ADJ_TYPE*.

Setting Up Adjustment Type Profiles

Adjustment type profiles categorize your adjustment types into logical groups. When you link a profile to a contract type, you limit the type of adjustments to be linked to the contract type's contracts. The creation of adjustment profiles and their linkage to contract types prevents inappropriate adjustments from being linked to your contracts. More than one adjustment type profile may be linked to a contract type.

For example, you can create an adjustment type profile called `Miscellaneous Fees` and link to it the miscellaneous fee adjustment types. Then, you would link this profile to those contract types that are allowed to levy such fees.

Note: Bottom line. An adjustment can only be linked to a contract if its adjustment type is part of an adjustment type profile that is valid for the contract's contract type. If an adjustment type is not linked to a profile, it could never be levied.

To set up adjustment type profiles, open **Admin Menu, Adjustment Type Profile**.

Description of Page

Enter a unique **Adjustment Type Profile** and **Description** for the adjustment type profile.

Indicate the **Adjustment Types** that are part of the profile.

Examples Of Common Adjustment Profiles

The following table shows several classic adjustment profiles used by many organizations (we've displayed some attributes from the adjustment type in the following table to help make it more understandable):

Adjustment Profile	Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm
FEES	NSF	NSF revenue	Your NSF fee	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`
	LPC	Late payment charge revenue	Your LPC	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`

Adjustment Profile	Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm
	CONNECT	Connection charge revenue	Your connection charge	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`
MISCEXP	CUSTREL	Customer relationship expense	N/A	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`
XFER	TRANSBAL	Balance transfer clearing	N/A	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`
REFUND	REFUND	A/P clearing	N/A	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`
DPA	ADJCURR	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`
	SYNCCURR	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`
CHARITY	CHAR FIX	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`
BUDGET	BUDG ON	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`
	BUDG OFF	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`
	BUDG FIX	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`

Adjustment Profile	Adjustment Type	Typical Distribution Code	Default Amount	Financial Transaction Algorithm
	FIX PAY	Customer relationship expense	N/A	`Payoff Amt = Adj Amt / Current Amt = 0`
DEPOSIT	DEPOSBILL	N/A	Your standard deposit amount	`Payoff Amt = 0 / Current Amt = Adj Amt`
	DEPOSINT	Interest expense	N/A	`Payoff Amt = Adj Amt / Current Amt = Adj Amt`
	SYNCCURR	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`
	ADJCUR	N/A	N/A	`Payoff Amt = 0 / Current Amt = Adj Amt`

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ADJ_TYP_PROF*.

The Big Picture of Adjustment Approval

Some implementations require adjustments to be approved by one or more managers before they impact a customer's debt and the general ledger. For example, an adjustment used to rebate a credit balance may require managerial approval before the rebate is sent to the customer. The topics in this section describe how to set up the system to support the approval of adjustments.

Approval Is Controlled By Approval Profiles

An approval profile contains the rules that define if and how an adjustment is approved. If an adjustment type does not reference an approval profile, the related adjustments do not require third-party approval before they impact a customer's debt. If an adjustment type references an approval profile, the approval profile's approval hierarchy defines if the adjustment requires approval and who the authorized approvers are. For example, an approval profile can be configured with the following approval hierarchy:

- Adjustments < $0 require approval by the "credit approvers role"
- Adjustments >= $0 and <= $10 do not require approval
- Adjustments > $10 and <= $100 require the approval of a user that belongs to the "level 1 approvers role"
- Adjustments > $100 require two levels of approval: first a user that belongs to the "level 1 approvers role" must approve the adjustment; afterwards, the adjustment must be approved by a user that belongs to the "level 2 approvers role"

Note: Transfer adjustments. The term "transfer adjustment" refers to two adjustments that are used to transfer money between two contracts. The adjustment with the positive amount is considered to be the debit adjustment; the other adjustment is considered the credit adjustment. When a transfer adjustment requires approval, only one of the adjustments needs to be approved. You control whether the debit side or the credit side of a transfer adjustment is used to control the approval process when you set up the approval profile.

Approval Profiles Can Be Linked To Multiple Adjustment Types

Approval hierarchies are frequently the same for many adjustment types. The system allows an approval profile to be linked to multiple adjustment types to simplify the definition and maintenance of the rules over time.

Adjustments Created In Batch Are Not Approved

The system assumes that no approval is necessary for adjustments created by batch processes even those whose adjustment type references an approval profile.

Approval Inserts A Step Into An Adjustment's Lifecycle

The Lifecycle Of An Adjustment explains how an adjustment is transitioned from the **Freezable** state to the **Frozen** state when it should impact the general ledger and the customer's balance. If an adjustment's adjustment type references an approval profile, the user cannot freeze the adjustment directly. Rather, the user must submit the adjustment for approval when it's ready and only when the last applicable approver approves the adjustment will it become **Frozen**.

Note: Freeze during bill completion. You can configure the system to only freeze certain types of adjustments when the next bill is completed for the adjustment's account. When the last approver approves such adjustments, they remain in the **Freezable** state. When the next bill is completed for the account, these adjustment become **Frozen**. Such adjustments that have not been approved at the time of bill completion will remain in the **Freezable** state. Refer to *Preventing Contract Balances And The GL From Being Impacted Until Bill Completion* on page 123 for more information.

Approval Requests Manage and Audit The Approval Process

Users submit an adjustment for approval using a dedicated button on the Adjustment page. When an adjustment is submitted for approval, the system creates an "approval request". The approval request determines if the adjustment requires approval and, if so, the list of approvers. If the adjustment does not require approval, the approval request is updated to indicate such and the adjustment is **Frozen** immediately (if freezing is allowed prior to bill completion). If the adjustment requires approval, the approval request's state becomes **Approval In Progress** and the approver(s) are notified.

Note: Approval submission logic is customizable. The previous paragraph describes how the base-package works when an adjustment is submitted for approval. This logic resides in an algorithm that's plugged in on the **C1-AdjustmentApprovalProfile** business object in the **Determine Approval Requirements** system event. Your implementation can change this logic by developing a new algorithm and plugging it into this business object. If your logic is meant to supersede the base-package algorithm, remember to inactivate the base-package algorithm by adding an appropriate inactivation option to this business object.

To Do Entries Are Created To Notify Approvers

When an approval request detects an adjustment requires approval, it notifies the first approver by creating a To Do entry. The To Do entry is created using the To Do type and To Do role defined on the approval profile. All users who belong to the approving To Do role can see the entry. When a user drills down on an adjustment approval To Do entry, the Adjustments - Approval portal is opened. This portal contains summary information about the adjustment and the approval history of the adjustment. This portal is also where the user approves or rejects the adjustment.

When the first user in the To Do role approves an adjustment, the To Do entry is **Completed** and the approval request's audit log is updated. If there are no higher levels of approval required, the adjustment is **Frozen** (if freezing is allowed prior to bill completion) and the approval request is moved to the **Approved** state. If there are higher levels of approval required, a new To Do entry is created to the next To Do role in the approval hierarchy.

Note: To Do entries can create email. A To Do entry can be configured to create an email message for every user in the To Do role to inform the user(s) of new adjustments requiring their attention. Refer to To Do Entries May Be Routed Out Of The System for the details.

Monitoring and Escalating Approval Requests

The base-package is supplied with an algorithm that your implementation can use to monitor approval requests that have been waiting too long for approval. This algorithm can complete the current To Do entry and create a new one for a different role when the timeout threshold defined on the algorithm's parameters is exceeded. If you've configured the system to send email for approval, this algorithm can also send x reminder emails (where x is defined on the algorithm's parameters) before the approval request is escalated to the new To Do role. Refer to C1-APR-TMOUT for more information about this algorithm. If you plan to enable this functionality, plug-in your configured algorithm on the **Approval In Progress** state on the **C1-AdjustmentApprovalRequest** business object.

Rejecting Deletes The Adjustment

When an adjustment is being approved, anyone with access to the adjustment can reject it by using the Adjustments - Approval portal. Users other than the current approver are allowed to reject an adjustment to allow an "in process" an adjustment to be withdrawn.

When an adjustment is rejected, the following takes place:

- The user is prompted for a reject reason.
- The approval request's audit log is updated with the reject reason and the approval request is moved to the **Rejected** state.
- The adjustment is deleted.

Note: In addition, a To Do entry is created using the **C1-ADJRJ** To Do type. You can search for the To Do entry in the **To Do List** screen. On clicking the **Submitted Message** link corresponding to the To Do entry, the **Adjustment** screen appears where you can view the details of the account whose adjustment was rejected. You can then create a new adjustment for the account.

Designing Your Approval Profiles

The following points describe a recommended design process:

- Create logical groups of adjustment types where each group has the same monetary hierarchy and approvers. An approval profile will be required for each of these groups.
- The number of To Do types (if any) that need to be created is dependent on how the adjustment approval To Do entries should be organized on To Do lists. For example, if all approval request To Do entries can appear in the same To Do list, you can use the base-package adjustment approval To Do type. However, if your organization prefers each approval profile's To Do entries to appear in a distinct To Do list, a separate To Do type will be needed for each list. Note that the base-package is supplied with a To Do type called C1-ADAPP that should be used as the basis for any new approval request To Do type.
- The number of To Do roles is dependent on who approves your adjustments. At a minimum, you will require a separate To Do role for each level in your approval profiles. Remember that every user in a To Do role will see its entries (and receive email if you've configured the system to do such).
- Refer to *Monitoring and Escalating Approval Requests* on page 155 for how to configure the system to escalate approval requests that have been waiting too long.
- If your implementation requires email notification when an adjustment requires approval, the following setup is required:
    - Set up an outbound message type, external system, and XAI sender. Refer to *To Do Entries May Be Routed Out Of The System* for the details.
    - Every To Do type referenced on your approval profiles should be configured as follows:

- Define the F1-TDEER batch process as the To Do type's routing process
- Set up an algorithm that references the C1-ADJAREQEM algorithm type and plug it in the **External Routing** system event.

Exploring Adjustment Approval Data Relationships

Use the following links to open the application viewer where the physical tables and data relationships behind the approval functionality are documented:

- Click C1-APPR PROF to view the approval profile maintenance object's tables.
- Click C1-APPR REQ to view the approval request maintenance object's tables.

Implementing Other Approval Paradigms

The above sections describe how the base-package adjustment approval process works. Because adjustment approval has been implemented using the **C1-AdjustmentApprovalProfile** and the **C1-AdjustmentApprovalRequest** business objects, your implementation can add additional business rules and change the approval user interface as required. Alternatively, if your implementation has a radically different approval process, you can create a different business objects with their own business rules. To learn how to do this, please enroll in the Configuration Tools training class.

Setting Up Approval Profiles

Approval profiles contain the rules that control how adjustments are approved. To set up an approval profile, open **Admin Menu**, **Approval Profile**.

Note: Refer to *The Big Picture of Adjustment Approval* on page 153 for a detailed description of how approval profiles govern the adjustment approval process.

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

Approval Profile List

The **Approval Profile List** zone lists the approval profiles that are already defined in the system. It contains the following columns:

Column Name	Column Description
Approval Profile	Displays the approval profiles defined in the system.
Description	Displays the description of the approval profile.

On clicking the **Broadcast** () icon corresponding to an approval profile, the **Approval Profile** and **Approval Profile's Adjustment Types** zone appears with the details of the respective approval profile.

Related Topics

For more information on...	See...
How to edit an approval profile	
How to delete an approval profile	
How to copy an approval profile	
How to view the details of an approval profile	

Approval Profile

The **Approval Profile** zone displays the details of an approval profile. It contains the following sections:

- **Main** - Displays basic information about the approval profile. It contains the following columns:

Column Name	Column Description
Approval Profile	Indicates the approval profile.
Description	Displays the description of the approval profile.
To Do Type	Indicates the type of To Do that is created for the approval profile. **Note:** It has a link. On clicking the link, the **To Do Type** screen appears where you can view the details of the respective To Do type.
Transfer Adjustment Precedence	Indicates the type of transfer adjustment precedence used for the approval profile.
Currency Conversion Required	Indicates whether currency conversion is required for the approval profile.

- **Debit Hierarchy** - Displays information about the debit hierarchy used in the approval profile. It contains the following columns:

Column Name	Column Description
Threshold Amount	Displays the threshold amount used while defining the debit hierarchy for the approval profile.
To Do Role	Displays the To Do role defined for the approval profile. **Note:** It has a link. On clicking the link, the **To Do Role** screen appears where you can view the details of the respective To Do role.

- **Credit Hierarchy** - Displays information about the credit hierarchy used in the approval profile. It contains the following columns:

Column Name	Column Description
Threshold Amount	Displays the threshold amount used while defining the credit hierarchy for the approval profile.
To Do Role	Displays the To Do role defined for the approval profile. **Note:** It has a link. On clicking the link, the **To Do Role** screen appears where you can view the details of the respective To Do role.

In addition to the above columns, this screen contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the approval profile.
Delete	Used to delete the approval profile.
Duplicate	Used to create a copy of the approval profile.

Related Topics

For more information on...	See...
How to edit an approval profile	<< adding links soon .... >>

For more information on...	See...
How to delete an approval profile	<< adding links soon .... >>
How to copy an approval profile	<< adding links soon .... >>
How to view the details of an approval profile	<< adding links soon .... >>

Approval Profile's Adjustment Types

The **Approval Profile's Adjustment Types** zone displays those adjustment types that make use of an approval profile. It contains the following columns:

Column Name	Column Description
Sr.No.	Displays the sequential number for every adjustment type.
Adjustment Type	Displays the adjustment type. **Note:** It contains a link. On clicking the link, the **Adjustment Type** screen appears where you can view the details of the respective adjustment type.

Note: By default, the **Approval Profile's Adjustment Types** zone does not appear in the **Approval Profile** screen. It appears only when the adjustment type is mapped with an approval profile.

Related Topics

For more information on...	See...
How to view the details of an approval profile	<< links will appear shortly >>

Designing and Defining Budget Plans

If you allow your customers to pay a budget amount each month (as opposed to their actual bill amount), you must set up one or more budget plans. If your company does not offer budget billing options, you can skip this section.

The topics in this section describe how to design and set up budget plans.

The Financial Impact Of Budget Plans

The only difference between a customer who participates in budget billing and one who doesn't is that budget billing customer have bill segments where payoff amount differs from current amount. Why? Because the payoff amount is the actual amount of the bill. The current amount is the amount the customer is expected to pay (i.e., their budget amount).

Let's run through an example of a customer on a budget to illustrate a contract where these two balances are not the same. The values in the payoff balance and current balance columns reflect the amount due after the financial transaction has been applied:

Date	Financial Transaction	Payoff Balance	Current Balance
1-Jan-99	Bill: $125, Budget $150	125	150
15-Jan-99	Payment: $150	-25	0
2-Feb-99	Bill: $175, Budget $150	150	150

Date	Financial Transaction	Payoff Balance	Current Balance
14-Feb-99	Payment: $150	0	0
3-Mar-99	Bill: $200, Budget $150	200	150
15-Mar-99	Payment: $150	50	0

Fastpath: For more information about current and payoff amounts, refer to *Current Amount versus Payoff Amount* .

What Do Budget Plans Do?

A budget plan contains the business rules that govern:

- How the recommended budget amount is calculated.
- When and how a customer on an ongoing budget plan will have their budget amount periodically trued up.
- The conditions under which the system will highlight an existing budget amount as being anomalous with the customer's current use patterns.

You may have different budget plans for different customer segments. For example, customers with large bills may have their budget amount recalculated every month, whereas small customers may have their budget amount only recalculated annually. You define which budget plans govern a customer's bills via a **budget plan on the customers' accounts**. An account's initial budget plan is defaulted from its customer class. You may override an account's budget plan at will.

Designing Your Budget Plans

Fastpath: Refer to *Budget Billing* for background information about budget billing.

A budget plan contains the business rules that govern:

- How the recommended budget amount is calculated.
- When and how a customer on an ongoing budget plan will have their budget amount periodically trued up.
- The conditions under which the system will highlight an existing budget amount as being anomalous with the customer's current use patterns.

You will need multiple budget plans if any of the above rules / conditions differ for different types of customers. For example, if residential customers use a different recommended budget algorithm as compared to commercial customers, you'd need one budget plan for residential and another for commercial.

We'll design budget plans to satisfy the needs of a theoretical company to help you understand how to design your budget plans. The following points describe the budget requirements of our theoretical company:

- The recommended budget amount is the last year's real bill amounts plus any existing debit/credit balance divided by 12. This is true regardless of the type of customer.
- The frequency of budget true up is monthly for commercial customers and annually for residential customers.
- The system should highlight when a residential customer's budget is more than 30% out of whack with what their budget amount would be if it was recalculated.
- The system should highlight when a commercial customer's budget is more than 20% out of whack with what their budget amount would be if it was recalculated.

You'd need the following budget plans to satisfy the above requirement:

Budget plan	Recommended Amount Algorithm	True Up Algorithm	Monitor Algorithm
`Residential`	Average Bill	True up every 12 months	Highlight when more than 30% out
`Commercial`	Average Bill	True up every month	Highlight when more than 20% out

Refer to the Page Controls under *Setting Up Budget Plans* for a description of the various algorithms defined in respect of a budget plan.

Setting Up Budget Plans

In the previous section, Designing Your Budget Plans, we presented a case study that illustrated a mythical organization's budget plans. In this section, we explain how to maintain your Budget Plans.

Budget Plan - Main

To set up budget plans, select **Admin Menu, Budget Plan**.

Description of Page

Enter an easily recognizable **Budget Plan** and **Description**.

The remaining information on this page is used by the various budget-oriented processes.

Refer to *Budget Plan - Calculation Algorithm* for information about the **Calculation Algorithm**.

Refer to *Budget Plan - Monitor Algorithm* for information about the **Monitor Algorithm**.

Refer to *Budget Plan - True Up Algorithm* for information about the **True Up Algorithm** and **Months for True Up**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_BUD_PLAN*.

Budget Plan - Calculation Algorithm

On *Budget Plan - Main* you must define the **Calculation Algorithm** used by the system when it calculates a customer's recommended budget amount.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that calculates recommended budget amounts. Click *here* to see the algorithm types available for this plug-in spot.

Budget Plan - Monitor Algorithm

On *Budget Plan - Main* you must define the **Monitor Algorithm** used by the *Budget Monitor* background process when it determines if a customer's budget plan is out-of-sync with their consumption patterns.

Note: What happens? If the algorithm determines that a customer's budget plan is out-of-sync with its current recommended amount, an entry is added to the *Budget Review* page.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that highlights if a customer's current budget amount is out-of-sync with their consumption patterns. Click *here* to see the algorithm types available for this plug-in spot.

Budget Plan - True Up Algorithm

On *Budget Plan - Main* you must define the **True Up Algorithm** used by the *Budget True Up* background process when it periodically trues up a customer's budget.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that trues up budget amounts. Click *here* to see the algorithm types available for this system event.

The system will automatically true up a customer's budget amount every X months (X is defined in **Months for True Up**).

Tender Management

When a payment is received, a tender is created to record what was remitted (e.g., cash, check, credit card). The topics in this section describe control tables that must be set up in order to remit tenders.

Fastpath: We strongly recommend *Tender Management and Workstation Cashiering* before setting up the control tables described in this section.

Setting Up Tender Types

Tender types are used to indicate the method in which the tender was made. A unique **Tender Type** must exist for every type of tender that can be remitted. For example, if you allow cash, checks, direct debits from a checking account, and direct debits from a credit card to be tendered, you'd need the following tender types:

Tender Type	Description	Like Cash	Generate Auto Pay	Require External Source ID	Require Expiration Date	External Type
CASH	Cash	Yes	No	N/A	N/A	N/A
CHEC	Check	No	No	N/A	N/A	N/A
OVUN	Cash drawer - over/under	No	No	N/A	N/A	N/A
DDCH	Direct debit - checking	No	Yes	Yes	No	`Checking withdrawal`
CRED	Direct debit - credit card	No	Yes	No	Yes	`Credit card withdrawal`

Go to **Admin Menu, Tender Type** to define your tender types.

Description of Page

Enter a unique **Tender Type** and **Description** for the tender type.

Turn on the **Like Cash** switch if this tender type is cash or the equivalent of cash. This indicator controls if the system generates a warning if a cash-only account remits a tender other than cash. It is also used to generate a warning for

online cashiers to turn in their tenders when the cash-like amount exceeds the maximum amount balance defined for the *tender source*.

Turn on **Generate Auto Pay** if this type of tender causes an automatic payment request to be routed to a financial institution. For example, this switch will be on if this tender type is used for direct debits from a customer's checking account (because every tender of this type will have an automatic payment request created when the tender is created).

The following fields are only used for tender types associated with automatic payments:

External Type This field is used by the background process that creates the information that is interfaced to the automatic payment source. Specifically, it controls the record type associated with the different types of automatic payments that are routed to the automated clearinghouse (ACH).

Note: The values for this field are customizable using the Lookup table. This field name is EXT_TYPE_FLG.

Require Ext. Src. ID This switch indicates if an Auto-Pay Source that references this type of tender must contain an External Source ID. The External Source ID is the unique identifier of the financial institution to which the automatic payment will be routed.

This switch is typically turned on for tender types associated with checking / saving direct debits. It is turned off for tender types associated with credit card debits (you don't need an external source for a credit card debit, you just need the credit card number).

Expiration Date Required Turn this switch on if an Auto-Pay Option that references an auto-pay source that references this type of tender must also contain an expiration date (e.g., automatic debit / credit cards).

Turn this switch off for tender types associated with checking / saving direct debits.

Tender Authorization Indicates that tenders of a particular type require authorization prior to being created.

Business Object If **Tender Authorization** has a value of `Required`, a **Business Object** must be specified for the tender type. The primary function of this **Business Object** is to manage the authorization of payment tenders.

Fastpath: For more information on authorizing credit card payments, refer to the `Tender Type - Credit Card with Authorization` business object.

Turn on **Allow Cash Back** if the system should automatically calculate a cash back amount when a tender is remitted for this tender type and the amount tender exceeds the amount being paid.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_TENDER_TYPE*.

Setting Up Tender Sources

A unique **Tender Source** must exist for every potential source of funds. For example,

- Every cashiering station will have a unique tender source.
- Every lock box will have a unique tender source.
- Your remittance processor will have a unique tender source.
- If you allow customers to pay bills automatically (e.g., via EFT), you'll need a tender source for each institution to which you route automatic payment requests. For example, if you route automatic payment requests to the automated clearinghouse (ACH), you'll need a tender source for the ACH.

For example, if you have 3 lock boxes, 2 cash drawers at an area office A, 2 cash drawers at area office B, and a single remittance processor, you'd need the following tender sources:

Tender Source	Type	External Source ID (Lockbox ID)	Default Starting Balance	Currency Code	Suspense Contract
CASH-A01	Cashiering	N/A	150.00	USD	N/A
CASH-A02	Cashiering	N/A	150.00	USD	N/A
CASH-B01	Cashiering	N/A	150.00	USD	N/A
CASH-B02	Cashiering	N/A	150.00	USD	N/A
LB-INDUS	Lockbox	112910-A	N/A	USD	9291019281
LB-COMM	Lockbox	938219-C	N/A	USD	4739837372
LB-RESID	Lockbox	372829-B	N/A	USD	1912910192
REMIT	Lockbox	N/A	N/A	USD	1920038437
ACH	Auto Pay	N/A	N/A	USD	N/A

To set up a tender source, select **Admin Menu, Tender Source**.

Description of Page

Enter an easily recognizable **Tender Source** and **Description** for the tender source.

Define the **Tender Source Type**. Valid values are: `Ad Hoc`, `Auto Pay`, `Online Cashiering` and `Lockbox`. The system uses this information to prevent tender controls from different sources from being included under the same deposit control. In other words, you can't mix ad hoc, automatic payment, cashiering and lockbox tenders under the same deposit control.

Fastpath: For more information, refer to *Maintaining Deposit Controls*.

If the source is an external system (e.g., a lockbox or an automatic payment destination), use **External Source ID** to define the unique identifier of the source. The background process that interfaces tenders from this source uses this information to create the appropriate tender control when it interfaces payments from external sources.

If this source is a cash drawer, define the **Default Starting Balance**. This balance is defaulted onto new tender controls and may be overridden.

Note: The tender type of the **Start Balance** is defined on the installation record.

If this source is a cash drawer, define the **Max Amount Balance**. When the amount of *cash-like* tenders in a cash drawer exceeds this balance, a warning is issued to remind the cashier to turn in some of the funds to a tender control.

Define the **Currency Code** of tenders linked to this source. All tenders in a source must be of the same currency.

If this tender source is associated with payments that are *interfaced from an external source* (e.g., a lockbox or a remittance processor), use Suspense **Contract** to define the contract whose account will hold uploaded payments with an invalid account. Refer to *Payment Upload Error Segmentation* for more information about suspense contracts. Also note, because the payment upload process simply books payments that reference invalid accounts to the account associated with this contract, this account should belong to a customer class with the appropriate payment distribution algorithms. This may entail creating a new customer class that will only be used on these "suspense accounts". This customer class would need the following algorithms:

- We'd recommend using a simple payment distribution algorithm like *PYDIST-PPRTY* (distribute payment based on contract type's payment priority).

- We'd recommend using an overpayment distribution algorithm like *OVRPY-PPRTY* (distribute overpayment to highest priority contract type).

Define the **Bank Code** and **Bank Account** into which the tender source's moneys will be deposited. The bank account defines the distribution code used to build the GL details for the payment. Refer to The *Source of GL Accounts on Financial Transactions* for more information. Note that the bank code and bank account can later be overwritten when entering Tender Deposits on *Deposit Control*.

If this tender source is associated with payments that are *interfaced from an external source*, for example tender sources associated with `Auto Pay` and `Lockbox` **Tender Source Types**, the information is also used as follows:

- The *payment upload process* uses this information to populate the bank and bank account when it creates deposit control records for the tender controls it creates during the interface. Refer to *Managing Payments Interfaced From External Sources* for more information.
- The *automatic payment interface* uses this information to populate the bank and bank account when it creates deposit control records for the tender controls it creates during the interface.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_TNDR_SRCE*.

Automatic Payment Options

If your customers can pay their bills automatically (via direct debit or credit card debits), you'll need to set up the various control tables described in this section.

Note: Important! Besides the tables described in this section, additional values must also be added to control tables defined under *Tender Management*. Specifically, refer to *Setting Up Tender Types* and *Setting Up Tender Sources*.

Fastpath: Refer to *Automatic Payments* for more information about how automatic payments are handled in the system.

Setting Up Auto-Pay Source Codes

A unique **Auto-Pay Source** must exist for every bank / credit card company / bill payment service that your customer's use as the source of the funds when they sign up for automatic payment. For example,

- Every bank will have a unique auto-pay source.
- Every credit card company will have a unique auto-pay source.

To set up an auto-pay source, select **Admin Menu, Auto Pay Source Type**.

Description of Page

Enter an easily recognizable **Auto Pay Source Code** and **Description** for the auto-pay source.

The **Source Name** is the name of the financial institution.

When the system creates an automatic payment request, it also creates an associated payment tender. This tender (like all tenders) must have a tender type. This field defines the **Tender Type** associated with this auto-pay source's tenders. Refer to *Setting Up Tender Types* for more information.

The **External Source ID** is the unique identifier of the financial institution to which the automatic payment will be routed (e.g., the bank routing ID of the bank). This field is typically blank on automatic payments routed to credit card companies because the credit card company doesn't have an external source ID (whereas direct debits from banks must have a bank routing number). Whether this field is required is controlled by the **Tender Type**.

The **Work Calendar** defines the financial institution's workdays. This information is used to determine the date on which automatic payment requests will be sent to the financial institution. Refer to *Setting Up External Workday Calendars* for more information.

The **Validation Algorithm** defines how the system validates the customer's account ID at the financial institution. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that validates the customer's account ID at the financial institution. Click *here* to see the algorithm types available for this plug-in spot.

Refer to *Account - Auto Pay* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_APAY_SRC*.

Auto Pay Route Type

An auto pay route type controls when and how automatic payment requests are routed to a financial institution, and when the general ledger is impacted. Each auto pay route type must be associated to a tender source. The system creates a new tender control each time it routes automatic payments to the tender source. Refer to *Managing Payments Interfaced From External Sources* for more information about tender source and tender control. The payment tenders of the automatic payments are then linked to the tender control for audit and control purposes.

The **Auto Pay Route Type** screen allows you to define, edit, copy, and delete an auto pay route type. It contains the following zones:

- *Auto Pay Route Type List* on page 165
- *Auto Pay Route Type* on page 166

Auto Pay Route Type List

The **Auto Pay Route Type List** zone lists the auto pay route types that are already defined in the system. It contains the following columns:

Column Name	Column Description
Auto Pay Route Type	Displays the auto pay route type.
Description	Displays the description of the auto pay route type.
Tender Source	Indicates the tender source through which the automatic payment must be remitted. **Note:** It has a link. On clicking the link, the **Tender Source** screen appears where you can view the details of the tender source.
Extract Batch	Indicates the batch which must be used to extract automatic payment and refund clearing records in a flat file. **Note:** It has a link. On clicking the link, the **Batch Control** screen appears where you can view the details of the batch.
Auto Pay Date Calculation Algorithm	Indicates the algorithm which must be used to calculate the extract date, distribution date, and payment date during the bill completion. **Note:** It has a link. On clicking the link, the **Algorithm** screen appears where you can view the details of the algorithm.

Column Name	Column Description
Edit	On clicking the **Edit** ( ) icon, the **Auto Pay Route Type** screen appears where you can edit the details of the auto pay route type.
Duplicate	On clicking the **Duplicate** ( ) icon, the **Auto Pay Route Type** screen appears where you can define a new auto pay route type using an existing auto pay route type.
Delete	On clicking the **Delete** ( ) icon, you can delete the auto pay route type. **Note:** You can delete an auto pay route type only when you have not created an auto pay instruction using the auto pay route type.

On clicking the **Broadcast** ( ) icon corresponding to an auto pay route type, the **Auto Pay Route Type** zone appears with the details of the respective auto pay route type.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165
Auto Pay Route Type zone	*Auto Pay Route Type* on page 166
How to edit an auto pay route type	*Editing an Auto Pay Route Type* on page 170
How to delete an auto pay route type	*Deleting an Auto Pay Route Type* on page 172
How to view the details of an auto pay route type	*Viewing the Auto Pay Route Type Details* on page 175
How to copy an auto pay route type	*Copying an Auto Pay Route Type* on page 173

Auto Pay Route Type

The **Auto Pay Route Type** zone displays the details of the auto pay route type. It contains the following sections:

- **Main** - Displays basic information about the auto pay route type. It contains the following fields:

Field Name	Field Description
Auto Pay Route Type	Displays the auto pay route type.
Description	Displays the description of the auto pay route type.
Detailed Description	Displays additional information about the auto pay route type.
Tender Source	Indicates the tender source through which the automatic payment must be remitted. **Note:** It has a link. On clicking the link, the **Tender Source** screen appears where you can view the details of the tender source.
Extract Batch	Indicates the batch which must be used to extract automatic payment and refund clearing records in a flat file. **Note:** It has a link. On clicking the link, the **Batch Control** screen appears where you can view the details of the batch.

Field Name	Field Description
Auto Pay Date Calculation Algorithm	Indicates the algorithm which must be used to calculate the extract date, distribution date, and payment date during the bill completion. **Note:** It has a link. On clicking the link, the **Algorithm** screen appears where you can view the details of the algorithm.

- **Auto Pay Characteristics** - Lists characteristics associated with the auto pay route type which you can define while creating an auto pay instruction using the auto pay route type. It contains the following columns:

Column Name	Column Description
Required	Indicates whether the characteristic is mandatory while creating an auto pay instruction using the auto pay route type.
Default	Indicates whether the default value is defined for the characteristic type.
Characteristic Type	Indicates the characteristic type.
Characteristic Value	Displays the default value of the characteristic type.

- **Record Actions** - This section contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the auto pay route type.
Delete	Used to delete the auto pay route type. **Note:** You can delete an auto pay route type only when you have not created an auto pay instruction using the auto pay route type.
Duplicate	Used to create a new auto pay route type using an existing auto pay route type.

- **Record Information** - This section contains the following field:

Field Name	Field Description
Business Object	Indicates the business object using which the auto pay route type is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.

By default, the **Auto Pay Route Type** zone does not appear in the **Auto Pay Route Type** screen. It appears only when you click the **Broadcast** ( ) icon corresponding to an auto pay route type in the **Auto Pay Route Type List** zone.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165
Auto Pay Route Type List zone	*Auto Pay Route Type List* on page 165
How to edit an auto pay route type	*Editing an Auto Pay Route Type* on page 170
How to delete an auto pay route type	*Deleting an Auto Pay Route Type* on page 172

For more information on...	See...
How to view the details of an auto pay route type	*Viewing the Auto Pay Route Type Details* on page 175
How to copy an auto pay route type	*Copying an Auto Pay Route Type* on page 173

Defining an Auto Pay Route Type

Prerequisites

To define an auto pay route type, you should have:

- Tender sources and extract batch control defined in the application
- Auto pay date calculation algorithm defined using the **APAY-DTCALC** algorithm type

Procedure

To define an auto pay route type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Auto Pay Route Type**.

   A sub-menu appears.

3. Click the **Add** option from the **Auto Pay Route Type** sub-menu.

   The **Auto Pay Route Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the auto pay route type.
   - **Auto Pay Characteristics** - Used to indicate the auto pay characteristic types that can be defined when you create an auto pay instruction using the auto pay route type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Route Type	Used to specify the auto pay route type.	Yes
Business Object	Indicates the auto pay route type business object using which you are defining the auto pay route type.	Not applicable
Description	Used to specify the description for the auto pay route type.	Yes
Detailed Description	Used to specify additional information about the auto pay route type.	No
Tender Source	Used to indicate the tender source through which the automatic payment must be remitted. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Tender Source Search** window appears.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Extract Batch	Used to indicate the batch that you want to use to extract automatic payment and refund clearing records in a flat file. **Note:** At present, the **Extract Automatic Payments (APAYACH)** batch extracts automatic payment clearing records with the latest batch run number in a flat file. It does not extract automatic refund clearing records in a flat file. The implementation team will have to develop the custom batch to extract both automatic payment and refund clearing records in a flat file. The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Batch Control Search** window appears.	Yes
Auto Pay Date Calculation Algorithm	Used to indicate the algorithm that you want to use to calculate the extract date, distribution date, and payment date during the bill completion. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Algorithm Search** window appears.	Yes

Tip: Alternatively, you can access this screen by clicking the **Add** button in the **Page Title** area of the **Auto Pay Route Type** screen.

4. Enter the required details in the **Main** section.
5. Associate auto pay characteristic types with the auto pay route type, if required.
6. Click **Save**.

   The auto pay route type is defined.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165
How to associate an auto pay characteristic type with an auto pay route type	*Associating an Auto Pay Characteristic Type with an Auto Pay Route Type* on page 169

Associating an Auto Pay Characteristic Type with an Auto Pay Route Type

Prerequisites

To associate an auto pay characteristic type with an auto pay route type, you should have:

- Characteristic types defined in the application (where the characteristic entity is set to **Auto Payment**)

Procedure

To associate an auto pay characteristic type with an auto pay route type:

1. Ensure that the
   Auto Pay Characteristics section is expanded when you are defining, editing, or copying an auto pay route type.

   The **Auto Pay Characteristics** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Required	Used to indicate whether the characteristic is mandatory while creating an auto pay instruction using the auto pay route type.	No
Default	Used to indicate whether you want to define the default value for the characteristic type.	No
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types where the characteristic entity is set to **Auto Payment**.	Yes (Conditional) **Note:** This field is required when you are associating an auto pay characteristic type with the auto pay route type.
Characteristic Value	Used to specify the default value for the characteristic type. **Note:** On specifying the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field.	No

2. Enter the required details in the **Auto Pay Characteristics** section.

   Note: If you select a predefined characteristic type, the **Search** () icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, you can search for a predefined characteristic value.

3. If you want to associate more than one auto pay characteristic type with the auto pay route type, click the **Add** () icon and then repeat step 2.

   Note: However, if you want to remove an auto pay characteristic type from the auto pay route type, click the **Delete** () icon corresponding to the auto pay characteristic type.

Related Topics

For more information on...	See...
How to define an auto pay route type	*Defining an Auto Pay Route Type* on page 168
How to edit an auto pay route type	*Editing an Auto Pay Route Type* on page 170
How to copy an auto pay route type	*Copying an Auto Pay Route Type* on page 173

Editing an Auto Pay Route Type

Prerequisites

To edit an auto pay route type, you should have:

- Tender sources and extract batch control defined in the application
- Auto pay date calculation algorithm defined using the **APAY-DTCALC** algorithm type

Procedure

To edit an auto pay route type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Auto Pay Route Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Auto Pay Route Type** sub-menu.

   The **Auto Pay Route Type** screen appears.

4. In the **Auto Pay Route Type List** zone, click the **Edit** () icon in the **Edit** column corresponding to the auto pay route type whose details you want to edit.

   The **Auto Pay Route Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the auto pay route type.
   - **Auto Pay Characteristics** - Used to indicate the auto pay characteristic types that can be defined when you create an auto pay instruction using the auto pay route type.

   The **Main** section contains the following fields:

Field Name	**Field Description**	**Mandatory (Yes or No)**
Auto Pay Route Type	Displays the auto pay route type.	Not applicable
Business Object	Indicates the auto pay route type business object used while defining the auto pay route type.	Not applicable
Description	Used to specify the description for the auto pay route type.	Yes
Detailed Description	Used to specify additional information about the auto pay route type.	No
Tender Source	Used to indicate the tender source through which the automatic payment must be remitted. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Tender Source Search** window appears.	Yes
Extract Batch	Used to indicate the batch that you want to use to extract automatic payment and refund clearing records in a flat file. **Note:** At present, the **Extract Automatic Payments (APAYACH)** batch extracts automatic payment clearing records with the latest batch run number in a flat file. It does not extract automatic refund clearing records in a flat file. The implementation team will have to develop the custom batch to extract both automatic payment and refund clearing records in a flat file. The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Batch Control Search** window appears.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Date Calculation Algorithm	Used to indicate the algorithm that you want to use to calculate the extract date, distribution date, and payment date during the bill completion. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Algorithm Search** window appears.	Yes

Tip: Alternatively, you can edit the details of an auto pay route type by clicking the **Edit** button in the **Auto Pay Route Type** zone.

5. Modify the required details in the **Main** section.
6. Associate, edit, or remove an auto pay characteristic type from the auto pay route type, if required.

Note: You cannot edit or remove an auto pay characteristic type from the auto pay route type when it is already used to define characteristic for an auto pay instruction.

7. Click **Save**.

   The changes made to the auto pay route type are saved.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165
Auto Pay Route Type List zone	*Auto Pay Route Type List* on page 165
Auto Pay Route Type zone	*Auto Pay Route Type* on page 166
How to associate an auto pay characteristic type with an auto pay route type	*Associating an Auto Pay Characteristic Type with an Auto Pay Route Type* on page 169

Deleting an Auto Pay Route Type

Procedure

To delete an auto pay route type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.
2. From the **Admin** menu, select **A** and then click **Auto Pay Route Type**.

   A sub-menu appears.
3. Click the **Search** option from the **Auto Pay Route Type** sub-menu.

   The **Auto Pay Route Type** screen appears.
4. In the **Auto Pay Route Type List** zone, click the **Delete**( ) icon in the **Delete** column corresponding to the auto pay route type that you want to delete.

   A message appears confirming whether you want to delete the auto pay route type.

Note: You can delete an auto pay route type only when you have not created an auto pay instruction using the auto pay route type.

> **Tip:** Alternatively, you can delete an auto pay route type by clicking the **Delete** button in the **Auto Pay Route Type** zone.

5. Click **OK**.

   The auto pay route type is deleted.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165
Auto Pay Route Type List zone	*Auto Pay Route Type List* on page 165
Auto Pay Route Type zone	*Auto Pay Route Type* on page 166

Copying an Auto Pay Route Type

Instead of creating an auto pay route type from scratch, you can create a new auto pay route type using an existing auto pay route type. This is possible through copying an auto pay route type. On copying an auto pay route type, the details including the auto pay characteristics are copied to the new auto pay route type. You can then edit the details, if required.

Prerequisites

To copy an auto pay route type, you should have:

- Auto pay route type (whose copy you want to create) defined in the application
- Tender sources and extract batch control defined in the application
- Auto pay date calculation algorithm defined using the **APAY-DTCALC** algorithm type

Procedure

To copy an auto pay route type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Auto Pay Route Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Auto Pay Route Type** sub-menu.

   The **Auto Pay Route Type** screen appears.

4. In the **Auto Pay Route Type List** zone, click the **Duplicate** () icon in the **Duplicate** column corresponding to the auto pay route type whose copy you want to create.

   The **Auto Pay Route Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the auto pay route type.
   - **Auto Pay Characteristics** - Used to indicate the auto pay characteristic types that can be defined when you create an auto pay instruction using the auto pay route type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Route Type	Used to specify the auto pay route type.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Business Object	Indicates the auto pay route type business object using which you are defining the auto pay route type.	Not applicable
Description	Used to specify the description for the auto pay route type.	Yes
Detailed Description	Used to specify additional information about the auto pay route type.	No
Tender Source	Used to indicate the tender source through which the automatic payment must be remitted. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Tender Source Search** window appears.	Yes
Extract Batch	Used to indicate the batch that you want to use to extract automatic payment and refund clearing records in a flat file. **Note:** At present, the **Extract Automatic Payments (APAYACH)** batch extracts automatic payment clearing records with the latest batch run number in a flat file. It does not extract automatic refund clearing records in a flat file. The implementation team will have to develop the custom batch to extract both automatic payment and refund clearing records in a flat file. The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Batch Control Search** window appears.	Yes
Auto Pay Date Calculation Algorithm	Used to indicate the algorithm that you want to use to calculate the extract date, distribution date, and payment date during the bill completion. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Algorithm Search** window appears.	Yes

Tip: Alternatively, you can create a copy of an auto pay route type by clicking the **Duplicate** button in the **Auto Pay Route Type** zone.

5. Enter the required details in the **Main** section.
6. Associate, edit, or remove an auto pay characteristic type from the auto pay route type, if required.
7. Click **Save**.

   The new auto pay route type is defined.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165

For more information on...	See...
Auto Pay Route Type List zone	*Auto Pay Route Type List* on page 165
Auto Pay Route Type zone	*Auto Pay Route Type* on page 166
How to associate an auto pay characteristic type with an auto pay route type	*Associating an Auto Pay Characteristic Type with an Auto Pay Route Type* on page 169

Viewing the Auto Pay Route Type Details

Procedure

To view the details of an auto pay route type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **A** and then click **Auto Pay Route Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Auto Pay Route Type** sub-menu.

   The **Auto Pay Route Type** screen appears.

4. In the **Auto Pay Route Type List** zone, click the **Broadcast** ( ) icon corresponding to the auto pay route type whose details you want to view.

   The **Auto Pay Route Type** zone appears.

5. View the details of the auto pay route type in the **Auto Pay Route Type** zone.

Related Topics

For more information on...	See...
Auto Pay Route Type screen	*Auto Pay Route Type* on page 165
Auto Pay Route Type List zone	*Auto Pay Route Type List* on page 165
Auto Pay Route Type zone	*Auto Pay Route Type* on page 166

Auto Pay Instruction (Used for Searching)

Until now, you were able to define an automatic payment option for an account from the **Auto Pay** tab of the **Account** screen. Now, in addition, you can define the following different types of the automatic payment options for an account:

- **Regular (i.e. Rule Based Auto Pay Instruction)** - An auto pay instruction which contains one or more rules. For more information, see *Rule Based Auto Pay Instructions*.
- **Default** - A default auto pay instruction is a non rule based auto pay instruction. You can only define one default auto pay instruction for the account within the given date range. The priority of the default auto pay instruction must be lowest compared to other rule based auto pay instructions. An effective default auto pay instruction is used when either none of the effective rule based auto pay instructions are satisfied on the bill's due date or when none of the rule based auto pay instructions are effective on the bill's due date.
- **Manual** - An auto pay instruction which should not be automatically used when the financial transactions of the account are frozen.

Note: You can define rule based, default, and manual auto pay instructions for an account only when the **Rule Based Auto Pay** check box in the **Auto Pay** tab of the **Account** screen is selected.

The **Auto Pay Instruction** screen allows you to search for an auto pay instruction using various search criteria. It also allows you to define rule based, default, and manual auto pay instructions. In addition, it allows you to edit, delete, and copy an auto pay instruction. It contains the following zone:

- *Search Auto Pay Instruction* on page 176

Through this screen, you can navigate to the following screen:

- *Auto Pay Instruction (Used for Viewing)* on page 188

Search Auto Pay Instruction

The **Search Auto Pay Instruction** zone allows you to search for a rule based, manual, and default auto pay instruction using various search criteria. It contains the following two sections:

- **Search Criteria** - The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Search By	Used to indicate whether you want to search for an auto pay instruction using the auto pay details. The valid value is: • Auto Pay Details **Note:** By default, the **Auto Pay Details** option is selected.	Yes
Auto Pay ID	Used to search a particular auto pay instruction.	No
Account ID	Used to search auto pay instructions of a particular account. **Note:** The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the **Account Search** window appears.	No
Auto Pay Route Type	Used to search auto pay instructions which are routed through a particular auto pay route type. **Note:** The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Route Type Search** window appears.	No
Start Date	Used to search auto pay instructions which are effective from a particular date onwards.	No
Auto Pay Source Code	Used to search auto pay instructions where the automatic payment request is sent to a particular financial institution. **Note:** The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Source Search** window appears.	No
End Date	Used to search auto pay instructions which are effective till a particular date. **Note:** If you do not specify the end date, the system sets the current date as the end date.	No

Field Name	Field Description	Mandatory (Yes or No)
Rule Type	Used to search rule based auto pay instructions which are created using a particular rule type.	No

Note: You must specify at least one search criterion while searching for an auto pay instruction.

- **Search Results** - On clicking the **Search** button, the search results appear based on the specified search criteria. The **Search Results** section contains the following columns:

Column Name	Column Description
Account Information	Indicates the account for which the auto pay instruction is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Account** screen appears where you can view the details of the respective account.
Priority	Indicates the priority in which the auto pay instruction should be considered when multiple auto pay instructions are effective on the bill's due date.
Auto Pay Type	Indicates the type of auto pay instruction. The valid values are: • Default • Manual • Regular
Auto Pay ID	Displays the auto pay ID. **Note:** It has a link. On clicking the link, the **Auto Pay Instruction** screen appears with the details of the respective auto pay instruction.
Auto Pay Source Code	Indicates the financial institution that receives the automatic payment request.
Auto Pay Route Type	Indicates when and how automatic payment request of the account is routed to a financial institution.
Account Number	Indicates the bank account number through which the automatic payment is made.
Start Date	Displays the date from when the auto pay instruction is effective.
End Date	Displays the date till when the auto pay instruction is effective.
Maximum Withdrawal Amount	Displays the maximum amount that can be automatically debited from the bank account.
Rule Type	Indicates the rule type using which the rule based auto pay instruction is created.

In addition, this section contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the automatic payment option. **Note:** You cannot edit the following details in the automatic payment option when the auto pay ID is already stamped on any financial transaction: • Auto Pay Type • Rule Type • Existing Rules and their Criteria
Delete	Used to delete the auto pay instruction. **Note:** You can delete an auto pay instruction of an account only when the auto pay ID is not yet stamped on any financial transaction.
Duplicate	Used to create a new auto pay instruction using an existing auto pay instruction.

Note: If you want to edit, delete, or copy an auto pay instruction, you must select the check box corresponding to the auto pay instruction and then click the respective button.

Related Topics

For more information on...	See...
How to search for an auto pay instruction	*Searching for an Auto Pay Instruction* on page 178
How to view the details of an auto pay instruction	*Viewing the Auto Pay Instruction Details* on page 179
How to edit an auto pay instruction	*Editing an Auto Pay Instruction* on page 190
How to delete an auto pay instruction	*Deleting an Auto Pay Instruction* on page 194
How to copy an auto pay instruction	*Copying an Auto Pay Instruction* on page 195

Searching for an Auto Pay Instruction

Prerequisites

To search for an auto pay instruction, you should have:

- Accounts, auto pay route types, auto pay sources, and rule types defined in the application

Procedure

To search for an auto pay instruction:

1. Click the **Menu** link in the **Application** toolbar.

   A list appears.

2. From the **Main** menu, select **Person Information** and then click **Auto Pay Instruction**.

   A sub-menu appears.

3. Click the **Search** option from the **Auto Pay Instruction** sub-menu.

   The **Auto Pay Instruction** screen appears.

4. Enter the search criteria in the **Search Auto Pay Instruction** zone.

Note: ORMB search engine supports wildcard search, where you can substitute the percentage (%) symbol as a stand in for any word or letter in a search criteria. You can use the '%' wildcard character in all input fields except the date and ID fields. The '%' wildcard character is suffixed automatically at the end of the partial search criteria. Therefore, you may or may not specify the wildcard character at the end of the partial search criteria. However, you have to prefix the wildcard character manually wherever required.

5. Click **Search**.

   A list of auto pay instructions that meet the search criteria appears in the **Search Results** section.

Related Topics

For more information on...	See...
Auto Pay Instruction screen	*Auto Pay Instruction (Used for Searching)* on page 175
Search Auto Pay Instruction zone	*Search Auto Pay Instruction* on page 176

Viewing the Auto Pay Instruction Details

Procedure

To view the details of an auto pay instruction:

1. Search for an auto pay instruction in the **Auto Pay Instruction** screen.
2. In the **Search Results** section, click the link in the **Auto Pay ID** column corresponding to the auto pay instruction whose details you want to view.

   The **Auto Pay Instruction** screen appears.
3. View the details of an auto pay instruction in the **Auto Pay Instruction** zone.

Related Topics

For more information on...	See...
How to search for an auto pay instruction	*Searching for an Auto Pay Instruction* on page 178
Auto Pay Instruction screen	*Auto Pay Instruction (Used for Viewing)* on page 188
Auto Pay Instruction zone	*Auto Pay Instruction* on page 188

Defining an Auto Pay Instruction

You can define an auto pay instruction for an account from the **Auto Pay Instruction** screen only when the **Rule Based Auto Pay** option is selected for the account.

Prerequisites

To define an auto pay instruction for an account, you should have:

- The **Rule Based Auto Pay** check box selected in the **Auto Pay** tab of the **Account** screen
- Auto pay sources and auto pay route types defined in the application
- Rule types defined in the application

Procedure

To define an auto pay instruction for an account:

1. Click the **Menu** link in the **Application** toolbar.

   A list appears.

2. From the **Main** menu, select **Person Information** and then click **Auto Pay Instruction**.

   A sub-menu appears.

3. Click the **Add** option from the **Auto Pay Instruction** sub-menu.

   The **Auto Pay Instruction** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the automatic payment option. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Account ID	Used to indicate the account for which you want to define the automatic payment option. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Account Search** window appears.	Yes
Rule Type	Used to indicate the rule type using which you want to create the rule based auto pay instruction.	Yes (Conditional) **Note:** This field is required when you select the **Regular** option from the **Auto Pay Type** list.

   - **Auto Pay Details** - Used to define the bank or credit card account from which the customer's automatic payments are debited. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Start Date	Used to specify the date from when the auto pay instruction is effective. **Note:** The start date cannot be later than the end date.	Yes
End Date	Used to specify the date till when the auto pay instruction is effective. **Note:** The end date cannot be earlier than the start date.	No

Field Name	Field Description	Mandatory (Yes or No)
Priority	Used to indicate the priority in which the auto pay instruction should be considered when multiple auto pay instructions are effective on the bill's due date. **Note:** The priority cannot be less than or equal to zero. The priority of the default auto pay instruction for the account must be lowest within the given date range. For example, there are two rule based auto pay instructions defined for 01-01-2017 to 31-12-2017 — one with the priority 10 and another with the priority 20. Now, if you want to define the default auto pay instruction for the account, you must define it with the priority 30 (which is lower than 10 and 20).	Yes
Auto Pay ID	Displays the auto pay ID. **Note:** The data appears in this field only when you are editing the details of an auto pay instruction.	Not applicable
Auto Pay Source Code	Used to indicate the financial institution that receives the automatic payment request. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Source Search** window appears.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Type	Used to indicate the type of auto pay instruction. The valid values are: • **Default** - Used when you want to create a default auto pay instruction for the account. A default auto pay instruction is a non rule based auto pay instruction. You can only define one default auto pay instruction for the account within the given date range. The priority of the default auto pay instruction must be lowest compared to other rule based auto pay instructions. An effective default auto pay instruction is used when either none of the effective rule based auto pay instructions are satisfied on the bill's due date or when none of the rule based auto pay instructions are effective on the bill's due date. • **Manual** - Used when you want to create auto pay instructions which should not be automatically used when the financial transactions of the account are frozen. • **Regular** - Used when you want to create a rule based auto pay instruction.	Yes
Auto Pay Route Type	Used to indicate when and how automatic payment request of the account is routed to a financial institution. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Route Type Search** window appears.	Yes
Auto Pay Method	Used to indicate how you want to process the automatic payment request. The valid values are: • Direct Debit • Payment Advice **Note:** This field appears only when the **Payment Advice Functionality Supported** option type in the **DISTBRULE** feature configuration is set to **Y**.	No
Account Number	Used to indicate the bank account number through which the automatic payment is made.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Expires On	Used to specify the date when the tender type of the auto pay source will expire. **Note:** The expiration date cannot be earlier than the start date.	Yes (Conditional) **Note:** This field is required when the **Expiration Date Required** check box is selected for the tender type.
Name	Used to specify the name of the person as it appears in the financial institution's system.	No
Maximum Withdrawal Amount	Used to specify the maximum amount that can be automatically debited from the bank account. It is used to set the limit on the withdrawal amount. **Note:** The maximum withdrawal amount cannot be less than zero.	No
Comments	Used to specify additional information about the automatic payment option.	No

- **Characteristics** - Used to define characteristics for the automatic payment option.
- **Rules** - Used to define one or more rules for a rule based auto pay instruction.

Note: This section appears only when you select a rule type in the **Main** section.

Tip: Alternatively, you can access this screen by clicking the **Add** button in the **Page Title** area of the **Auto Pay Instruction** screen.

4. Enter the required details in the **Main** and **Auto Pay Details** sections depending on whether you want to define a default, manual, or rule based auto pay instruction.
5. Define characteristics for the auto pay instruction, if required.
6. Define rules in the auto pay instruction when the **Auto Pay Type** is set to **Regular**.

Note:

You must define at least one rule when the **Auto Pay Type** field is set to **Regular**.

The number of rules defined in the auto pay instruction must not exceed the maximum rule count defined in the rule type.

7. Click **Save**.

   The auto pay instruction is defined.

Note: You cannot define multiple auto pay instructions for the account with the same priority and which are effective during the same date range.

Related Topics

For more information on...	See...
Auto Pay Instruction screen	*Auto Pay Instruction (Used for Searching)* on page 175
How to define characteristics for an auto pay instruction	*Defining Characteristics for an Auto Pay Instruction* on page 184

For more information on...	See...
How to define rules in an auto pay instruction	*Defining Rules in an Auto Pay Instruction* on page 185

Defining Characteristics for an Auto Pay Instruction

You can only define those characteristic types for an auto pay instruction which are associated with the auto pay route type used while creating the auto pay instruction. If the **Required** check box is selected while associating an auto pay characteristic type with the auto pay route type, you need to define the characteristic for the auto pay instruction. Otherwise, erroneous results might occur.

Prerequisites

To define characteristics for an auto pay instruction, you should have:

- Characteristic types (where the characteristic entity is set to **Auto Payment**) associated with the auto pay route type

Procedure

To define characteristics for an auto pay instruction:

1. Ensure that the **Characteristics** section is expanded when you are defining, editing, or copying an auto pay instruction.

   The **Characteristics** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types which are associated with the auto pay route type.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the auto pay instruction.
Characteristic Value	Used to specify the value for the characteristic type. **Note:** The default value, if defined while associating the auto pay characteristic type with the auto pay route type, appears when you select the characteristic type. You can edit the characteristic value, if required. On specifying the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the auto pay instruction.

2. Enter the required details in the **Characteristics** section.

   Note: If you select a predefined characteristic type, the **Search** () icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, you can search for a predefined characteristic value.

3. If you want to define more than one characteristic for the auto pay instruction, click the **Add** () icon and then repeat step 2.

   Note: However, if you want to remove a characteristic from the auto pay instruction, click the **Delete** () icon corresponding to the characteristic.

Related Topics

For more information on...	See...
How to define an auto pay instruction	*Defining an Auto Pay Instruction* on page 179
How to edit an auto pay instruction	*Editing an Auto Pay Instruction* on page 190
How to copy an auto pay instruction	*Copying an Auto Pay Instruction* on page 195

Defining Rules in an Auto Pay Instruction

You can define rules in an auto pay instruction only when the auto pay type is set to **Regular**. Once you select a rule type, you can define a rule criteria using the following:

- Fields which are selected as the input and output parameters in the rule type
- Characteristics of entities which are selected as the rule criteria characteristic entities in the rule type

You can define one or more rules in an auto pay instruction. Each rule contains one or more criteria. If a rule contains more than one criteria, each criteria is executed in the specified sequence. If all criteria defined in the rule are satisfied, the auto pay ID is stamped on the financial transaction during the **FT Freeze** system event. However, if the rule is not satisfied, each criteria in the next rule is executed in the specified sequence. If all rules defined in the auto pay instruction are not satisfied, the system checks whether the effective auto pay instruction with the next priority satisfies the financial transaction.

Prerequisites

To define rules in an auto pay instruction, you should have:

- Input and output parameters defined in the rule type
- Characteristic entities and rule criteria derivation algorithm associated with the rule type

Procedure

To define rules in an auto pay instruction:

1. Ensure that the **Rules** section is expanded when you are defining, editing, or copying an auto pay instruction.

   The **Rules** section contains the following field:

Field Name	Field Description	Mandatory (Yes or No)
Rule Description	Used to specify the description for the rule.	Yes

   In addition, this section contains the **Criteria** sub-section where you can define the rule criteria. The **Criteria** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Sequence	Indicates the order in which you want to execute the rule criteria.	Not applicable
Criteria Type	Used to indicate whether you want to define the criteria using a field or characteristic type. The valid values are: • Field • Characteristic	Yes

Field Name	Field Description	Mandatory (Yes or No)

Criteria Details	Used to define the rule criteria. If the criteria type is set to **Field**, it contains the following sub-fields:
	• **First** - Used to indicate the field using which you want to define the criteria. At present, the following fields are only supported while defining a rule based auto pay instruction — **Policy Number**, **Plan Number**, and **Price Item**.
	• **Second** - Used to indicate the operator that you want to use in the criteria. The valid values are:
	  • <
	  • <=
	  • <>
	  • =
	  • >
	  • >=
	  • Between
	  • In
	  • Like
	• **Third** - Used to specify the field value in the criteria.
	Note:
	The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the respective search window appears.
	The **Search** ( ) icon appears only when the search zone is selected for the respective field while defining input and output parameters for the rule type.
		Yes

Field Name	Field Description	Mandatory (Yes or No)

	If the criteria type is set to **Characteristic**, it contains the following sub-fields:
	• **First** - Used to indicate the entity whose characteristic types you want to use while defining the criteria. At present, the following characteristic entities are only supported while defining a rule based auto pay instruction — **Adjustment**, **Billable Charge**, **Policy**, **Policy Plan**, and **Price Item**.
	• **Second** - Used to indicate the characteristic type using which you want to define the criteria. The characteristic types listed change depending on the characteristic entity that you have selected.
	• **Third** - Used to indicate the operator that you want to use in the criteria. The valid values are:
	  • <
	  • <=
	  • <>
	  • =
	  • >
	  • >=
	  • Between
	  • In
	  • Like
	• **Fourth** - Used to specify the characteristic value in the criteria.
	Note: If you select a predefined characteristic type, the **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, you can search for a predefined characteristic value.	

2. Enter the rule description in the **Rules** section.
3. Enter the required details in the **Criteria** section to define the rule criteria.
4. If you want to define more than one criteria in the rule, click the **Add** ( ) icon in the **Criteria** section and then repeat step 3.

Note: However, if you want to remove a criteria from the rule, click the **Delete** ( ) icon corresponding to the criteria.

5. If you want to define more than one rule in the auto pay instruction, click the **Add** ( ) icon in the **Rules** section and then repeat step 2 and 3.

Note: However, if you want to remove a rule from the auto pay instruction, click the **Delete** ( ) icon corresponding to the rule.

Related Topics

For more information on...	See...
How to define an auto pay instruction	*Defining an Auto Pay Instruction* on page 179
How to edit an auto pay instruction	*Editing an Auto Pay Instruction* on page 190
How to copy an auto pay instruction	*Copying an Auto Pay Instruction* on page 195

Auto Pay Instruction (Used for Viewing)

The **Auto Pay Instruction** screen allows you to view the details of an automatic payment option of an account. It also allows you to edit, delete, and copy an auto pay instruction. It contains the following zone:

- *Auto Pay Instruction* on page 188

Auto Pay Instruction

The **Auto Pay Instruction** zone displays the details of the automatic payment option. It contains the following sections:

- **Main** - Displays basic information about the automatic payment option. It contains the following fields:

Field Name	Field Description
Auto Pay ID	Displays the auto pay ID.
Account Information	Indicates the account for which the automatic payment option is created. **Note:** It has a link. On clicking the link, the **Account** screen appears where you can view the details of the respective account.
Rule Type	Indicates the rule type using which the rule based auto pay instruction is created. **Note:** It has a link. On clicking the link, the **Rule Type** screen appears with the details of the respective rule type.

- **Auto Pay Details** - Displays the details of the bank or credit card account from which the customer's automatic payments are debited. It contains the following fields:

Field Name	Field Description
Start Date	Displays the date from when the auto pay instruction is effective.
End Date	Displays the date till when the auto pay instruction is effective.
Priority	Indicates the priority in which the auto pay instruction should be considered when multiple auto pay instructions are effective on the bill’s due date.
Auto Pay Source Code	Indicates the financial institution that receives the automatic payment request.
Auto Pay Route Type	Indicates when and how automatic payment request of the account is routed to a financial institution.
Auto Pay Method	Indicates how you want to process the automatic payment request. The valid values are: • Direct Debit • Payment Advice

Field Name	Field Description
Auto Pay Type	Indicates the type of auto pay instruction. The valid values are: • Default • Manual • Regular
Account Number	Indicates the bank account number through which the automatic payment is made.
Expires On	Displays the date when the tender type of the auto pay source will expire.
Name	Displays the name of the person as it appears in the financial institution's system.
Maximum Withdrawal Amount	Displays the maximum amount that can be automatically debited from the bank account.
Comments	Displays additional information about the automatic payment option.

- **Record Actions** - This section contains the following buttons:

Column Name	Column Description
Edit	Used to edit the details of the automatic payment option. **Note:** You cannot edit the following details in the automatic payment option when the auto pay ID is already stamped on any financial transaction: • Auto Pay Type • Rule Type • Existing Rules and their Criteria
Delete	Used to delete the auto pay instruction. **Note:** You can delete an auto pay instruction of an account only when the auto pay ID is not yet stamped on any financial transaction.
Duplicate	Used to create a new auto pay instruction using an existing auto pay instruction.

- **Record Information** - This section contains the following field:

Field Name	Field Description
Business Object	Indicates the business object using which the auto pay instruction is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.

- **Characteristics** - Lists the characteristics defined for the automatic payment option. It contains the following columns:

Column Name	Column Description
Characteristic Type	Indicates the characteristic type.
Characteristic Value	Displays the value of the characteristic type.

- **Rules** - Lists the rules defined in the rule based auto pay instruction. It contains the following field:

Field Name	Field Description
Rule Description	Displays the description of the rule.

In addition, this section contains the **Criteria** sub-section where you can view the criteria defined in the rule. The **Criteria** section contains the following columns:

Column Name	Column Description
Sequence	Displays the order in which the criteria should be executed in the rule.
Criteria Type	Indicates whether the criteria is defined using a field or characteristic type. The valid values are: • Field • Characteristic
Criteria Details	Displays the condition defined in the rule. It contains the field or characteristic type, operator, and field or characteristic value.

Related Topics

For more information on...	See...
Auto Pay Instruction screen	*Auto Pay Instruction (Used for Viewing)* on page 188
How to view the details of an auto pay instruction	*Viewing the Auto Pay Instruction Details* on page 179
How to edit an auto pay instruction	*Editing an Auto Pay Instruction* on page 190
How to delete an auto pay instruction	*Deleting an Auto Pay Instruction* on page 194
How to copy an auto pay instruction	*Copying an Auto Pay Instruction* on page 195

Editing an Auto Pay Instruction

Prerequisites

To edit an auto pay instruction of an account, you should have:

- Auto pay sources and auto pay route types defined in the application
- Rule types defined in the application

Procedure

To edit an auto pay instruction of an account:

1. Search for the automatic payment option of the account in the **Auto Pay Instruction** screen.
2. In the **Search Results** section, select the check box corresponding to the automatic payment option whose details you want to edit.
3. Click **Edit**.

   The **Auto Pay Instruction** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the automatic payment option. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Account ID	Indicates the account for which the automatic payment option is created.	Not applicable
Rule Type	Used to indicate the rule type using which you want to create the rule based auto pay instruction. **Note:** This field is editable only when the auto pay ID is not yet stamped on any financial transaction.	Yes (Conditional) **Note:** This field is required when you select the **Regular** option from the **Auto Pay Type** list.

- **Auto Pay Details** - Used to define the bank or credit card account from which the customer's automatic payments are debited. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Start Date	Used to specify the date from when the auto pay instruction is effective. **Note:** The start date cannot be later than the end date.	Yes
End Date	Used to specify the date till when the auto pay instruction is effective. **Note:** The end date cannot be earlier than the start date.	No
Priority	Used to indicate the priority in which the auto pay instruction should be considered when multiple auto pay instructions are effective on the bill's due date. **Note:** The priority cannot be less than or equal to zero. The priority of the default auto pay instruction for the account must be lowest within the given date range. For example, there are two rule based auto pay instructions defined for 01-01-2017 to 31-12-2017 — one with the priority 10 and another with the priority 20. Now, if you want to define the default auto pay instruction for the account, you must define it with the priority 30 (which is lower than 10 and 20).	Yes
Auto Pay ID	Displays the auto pay ID.	Not applicable

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Source Code	Used to indicate the financial institution that receives the automatic payment request. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Source Search** window appears.	Yes
Auto Pay Type	Used to indicate the type of auto pay instruction. The valid values are: • **Default** - Used when you want to create a default auto pay instruction for the account. A default auto pay instruction is a non rule based auto pay instruction. You can only define one default auto pay instruction for the account within the given date range. The priority of the default auto pay instruction must be lowest compared to other rule based auto pay instructions. An effective default auto pay instruction is used when either none of the effective rule based auto pay instructions are satisfied on the bill's due date or when none of the rule based auto pay instructions are effective on the bill's due date. • **Manual** - Used when you want to create auto pay instructions which should not be automatically used when the financial transactions of the account are frozen. • **Regular** - Used when you want to create a rule based auto pay instruction. **Note:** This field is editable only when the auto pay ID is not yet stamped on any financial transaction.	Yes
Auto Pay Route Type	Used to indicate when and how automatic payment request of the account is routed to a financial institution. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Route Type Search** window appears.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Method	Used to indicate how you want to process the automatic payment request. The valid values are: • Direct Debit • Payment Advice **Note:** This field appears only when the **Payment Advice Functionality Supported** option type in the **DISTBRULE** feature configuration is set to **Y**.	No
Account Number	Used to indicate the bank account number through which the automatic payment is made.	Yes
Expires On	Used to specify the date when the tender type of the auto pay source will expire. **Note:** The expiration date cannot be earlier than the start date.	Yes (Conditional) **Note:** This field is required when the **Expiration Date Required** check box is selected for the tender type.
Name	Used to specify the name of the person as it appears in the financial institution's system.	No
Maximum Withdrawal Amount	Used to specify the maximum amount that can be automatically debited from the bank account. It is used to set the limit on the withdrawal amount. **Note:** The maximum withdrawal amount cannot be less than zero.	No
Comments	Used to specify additional information about the automatic payment option.	No

- **Characteristics** - Used to define characteristics for the automatic payment option.
- **Rules** - Used to define one or more rules for a rule based auto pay instruction.

Note: This section appears only when you select a rule type in the **Main** section.

Tip: Alternatively, you can edit an auto pay instruction by clicking the **Edit** button in the **Auto Pay Instruction** zone.

4. Modify the required details in the **Main** and **Auto Pay Details** sections.
5. Define, edit, or remove characteristics from the auto pay instruction, if required.
6. Define, edit, or remove a criteria from the rule, if required.

Note: You cannot add, edit, or remove the criteria from the rule when the auto pay ID is already stamped on any financial transaction.

7. Define, edit, or remove rules from the rule based auto pay instruction, if required.

> **Note:**
>
> You must define at least one rule when the **Auto Pay Type** field is set to **Regular**.
>
> The number of rules defined in the auto pay instruction must not exceed the maximum rule count defined in the rule type.
>
> You cannot edit or remove the rule from the rule based auto pay instruction when the auto pay ID is already stamped on any financial transaction.

8. Click **Save**.

   The changes made to the auto pay instruction are saved.

> **Note:** You cannot define multiple auto pay instructions for the account with the same priority and which are effective during the same date range.

Related Topics

For more information on...	See...
How to search for an auto pay instruction	*Searching for an Auto Pay Instruction* on page 178
Auto Pay Instruction zone	*Auto Pay Instruction* on page 188
How to define characteristics for an auto pay instruction	*Defining Characteristics for an Auto Pay Instruction* on page 184
How to define rules in an auto pay instruction	*Defining Rules in an Auto Pay Instruction* on page 185

Deleting an Auto Pay Instruction

Procedure

To delete an auto pay instruction of an account:

1. Search for the automatic payment option of the account in the **Auto Pay Instruction** screen.
2. In the **Search Results** section, select the check box corresponding to the automatic payment option that you want to delete.
3. Click **Delete**.

   A message appears confirming whether you want to delete the auto pay instruction.

> **Note:** You can delete an auto pay instruction of an account only when the auto pay ID is not yet stamped on any financial transaction.

> **Tip:** Alternatively, you can delete an auto pay instruction by clicking the **Delete** button in the **Auto Pay Instruction** zone.

4. Click **OK**.

   The auto pay instruction is deleted.

Related Topics

For more information on...	See...
How to search for an auto pay instruction	*Searching for an Auto Pay Instruction* on page 178
Auto Pay Instruction zone	*Auto Pay Instruction* on page 188

Copying an Auto Pay Instruction

Instead of creating an auto pay instruction from scratch, you can create a new auto pay instruction using an existing auto pay instruction. This is possible through copying an auto pay instruction. On copying an auto pay instruction, the details including the characteristics and rules are copied to the new auto pay instruction. You can then edit the details, if required.

Prerequisites

To copy an auto pay instruction, you should have:

- Auto pay instruction (whose copy you want to create) defined in the application
- Auto pay sources and auto pay route types defined in the application
- Rule types defined in the application

Procedure

To copy an auto pay instruction:

1. Search for the automatic payment option in the **Auto Pay Instruction** screen.
2. In the **Search Results** section, select the check box corresponding to the automatic payment option whose copy you want to create.
3. Click **Duplicate**.

   The **Auto Pay Instruction** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the automatic payment option. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Account ID	Indicates the account for which the automatic payment option is created.	Not applicable
Rule Type	Used to indicate the rule type using which you want to create the rule based auto pay instruction.	Yes (Conditional) **Note:** This field is required when you select the **Regular** option from the **Auto Pay Type** list.

   - **Auto Pay Details** - Used to define the bank or credit card account from which the customer's automatic payments are debited. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Start Date	Used to specify the date from when the auto pay instruction is effective. **Note:** The start date cannot be later than the end date.	Yes
End Date	Used to specify the date till when the auto pay instruction is effective. **Note:** The end date cannot be earlier than the start date.	No

Field Name	Field Description	Mandatory (Yes or No)
Priority	Used to indicate the priority in which the auto pay instruction should be considered when multiple auto pay instructions are effective on the bill's due date. **Note:** The priority cannot be less than or equal to zero. The priority of the default auto pay instruction for the account must be lowest within the given date range. For example, there are two rule based auto pay instructions defined for 01-01-2017 to 31-12-2017 — one with the priority 10 and another with the priority 20. Now, if you want to define the default auto pay instruction for the account, you must define it with the priority 30 (which is lower than 10 and 20).	Yes
Auto Pay ID	Displays the auto pay ID. **Note:** The data appears in this field only when you are editing the details of an auto pay instruction.	Not applicable
Auto Pay Source Code	Used to indicate the financial institution that receives the automatic payment request. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Source Search** window appears.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Auto Pay Type	Used to indicate the type of auto pay instruction. The valid values are: • **Default** - Used when you want to create a default auto pay instruction for the account. A default auto pay instruction is a non rule based auto pay instruction. You can only define one default auto pay instruction for the account within the given date range. The priority of the default auto pay instruction must be lowest compared to other rule based auto pay instructions. An effective default auto pay instruction is used when either none of the effective rule based auto pay instructions are satisfied on the bill's due date or when none of the rule based auto pay instructions are effective on the bill's due date. • **Manual** - Used when you want to create auto pay instructions which should not be automatically used when the financial transactions of the account are frozen. • **Regular** - Used when you want to create a rule based auto pay instruction.	Yes
Auto Pay Route Type	Used to indicate when and how automatic payment request of the account is routed to a financial institution. **Note:** The **Search** () icon appears corresponding to this field. On clicking the **Search** icon, the **Auto Pay Route Type Search** window appears.	Yes
Auto Pay Method	Used to indicate how you want to process the automatic payment request. The valid values are: • Direct Debit • Payment Advice **Note:** This field appears only when the **Payment Advice Functionality Supported** option type in the **DISTBRULE** feature configuration is set to **Y**.	No
Account Number	Used to indicate the bank account number through which the automatic payment is made.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Expires On	Used to specify the date when the tender type of the auto pay source will expire. **Note:** The expiration date cannot be earlier than the start date.	Yes (Conditional) **Note:** This field is required when the **Expiration Date Required** check box is selected for the tender type.
Name	Used to specify the name of the person as it appears in the financial institution's system.	No
Maximum Withdrawal Amount	Used to specify the maximum amount that can be automatically debited from the bank account. It is used to set the limit on the withdrawal amount. **Note:** The maximum withdrawal amount cannot be less than zero.	No
Comments	Used to specify additional information about the automatic payment option.	No

- **Characteristics** - Used to define characteristics for the automatic payment option.
- **Rules** - Used to define one or more rules for a rule based auto pay instruction.

Note: This section appears only when you select a rule type in the **Main** section.

Tip: Alternatively, you can copy an auto pay instruction by clicking the **Duplicate** button in the **Auto Pay Instruction** zone.

4. Enter the required details in the **Main** and **Auto Pay Details** sections depending on whether you want to define a default, manual, or rule based auto pay instruction.
5. Define, edit, or remove characteristics from the auto pay instruction, if required.
6. Define, edit, or remove a criteria from the rule, if required.
7. Define, edit, or remove rules from the rule based auto pay instruction, if required.

Note:

You must define at least one rule when the **Auto Pay Type** field is set to **Regular**.

The number of rules defined in the auto pay instruction must not exceed the maximum rule count defined in the rule type.

8. Click **Save**.

   The new auto pay instruction is defined.

Note: You cannot define multiple auto pay instructions for the account with the same priority and which are effective during the same date range.

Related Topics

For more information on...	See...
How to search for an auto pay instruction	*Searching for an Auto Pay Instruction* on page 178
Auto Pay Instruction zone	*Auto Pay Instruction* on page 188

For more information on...	See...
How to define characteristics for an auto pay instruction	*Defining Characteristics for an Auto Pay Instruction* on page 184
How to define rules in an auto pay instruction	*Defining Rules in an Auto Pay Instruction* on page 185

Payment Advices

The topics in this section provide background information about payment advice functionality.

Note: This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization issues payment advices to the customer instead of initiating electronic funds transfer directly to the customer's bank.

What Is A Payment Advice?

Payment Advice is a note that is sent to the customer. It indicates the payment amount and the customer's bank details. If the customer agrees to the information on the payment advice, he/she signs it and returns it to the clearinghouse address that is indicated on the payment advice. The clearinghouse, in turn, sends the dated and signed payment advice to the customer's bank, which completes the payment.

Payment Advice vs. Direct Debit

The existing functionality that creates automatic payments is referred to as direct debit processing. Payment advice processing differs from direct debit processing in the way that automatic payments get initiated. With payment advice processing, the usual automatic payment records - i.e. payment event, payment, tender and auto pay clearinghouse staging - are not created. Instead, a payment advice is printed and sent to the customer. The customer sends the approved payment advice directly to the clearinghouse.

Note: The system does not provide sample processes for extracting and printing payment advice information. Your implementation team would have to create these.

Setting Up The System To Enable Payment Advices

You must set up the **Financial Transaction Options** feature configuration to define parameters that control payment advice functionality.

The following points describe the various **Option Types** that must be defined:

- `Payment Advice Functionality Supported.` This option controls whether the system allows for payment advice processing.
  - Enter `Y` if the system should allow for both direct debit and payment advice processing.
  - Enter `N` if the system should only allow for direct debit processing.
- `Default Auto Pay Method.` This option is used for defaulting the auto pay method on new account auto pay records.

Refer to *Account - Auto Pay* for more information on auto pay method.

Note: The system assumes direct debit processing if the above feature options are not defined.

Credit Card Payments

If your organization accepts credit card payments, you can configure the system to authorize customers' credit card charges in real-time, and perform an authorization reversal (also in real-time) when the credit card payment is canceled.

When the authorization web service is not available, you can permit users to enter authorization codes manually so that they can continue processing payments.

Configuring the System for Tender Authorization

The following sections describe the setup required if your organization intends to use the base CyberSource integration tender authorization functionality.

Define the Outbound Message Type

An outbound message type is required for the CyberSource authorization outbound message. This outbound message type must reference the base `CyberSource - Credit Card Authorization` business object.

An outbound message type is required for the CyberSource reversal outbound message. This outbound message type must reference the base `CyberSource - Credit Card Reversal` business object.

Define the XAI Sender

An XAI Sender is required to define how to send messages to CyberSource. Use the context of the XAI Sender to define the web service interface.

Define the External System and Configure the Messages

Define an external system and configure the valid outbound message types and the method of communication for each (`XAI, Batch, or Real Time`; `Real Time` is generally the appropriate choice for credit card authorization). You will also need to select the appropriate XSLs to format both the request and response to the outbound message types for CyberSource.

Define a User

Add a user to hold details required for CyberSource communication. Security information (e.g. Merchant Id, Merchant Reference Code, CyberSource User Name and Password) needed to interface with CyberSource is stored as user characteristics.

Set up the Tender Authorization Algorithm

A `Tender Type (BO) - Tender Authorization` algorithm must be configured. This algorithm performs a tender authorization or a tender authorization reversal through CyberSource.

Define a Business Object

A business object (BO) must be created for the `TENDER TYPE` maintenance object. This BO must reference the tender authorization algorithm created.

Define Tender Types

Update the appropriate tender type(s) to denote that authorization is required. The new business object must be specified on the tender type(s).

Tender Authorization - Feature Configuration

If your implementation has a need to prevent users from overriding the automatic tender authorization, then you must set up the `Allow Manual Tender Authorization Override` option type on the `Financial Transaction Options` *Feature Configuration*. The `Allow Manual Tender Authorization Override` option must have a value of `N` in order to suppress the Authorization Override checkbox.

Fastpath: For more information on credit card payment authorization refer to the `Tender Type - Credit Card with Authorization` business object.

Non-CIS Payments

Payment Templates can be configured for common types of non-CIS payment allocations. These templates are used to default the payment distribution and allow non-CIS payments to be directly allocated to specific distribution codes.

The section that follows provides an overview of the maintenance objects that support this functionality.

Payment Template

A payment template is an admin maintenance object (MO) used to define configuration rules for non-CIS payments. You can use a payment template to default the payment distribution for common types of non-CIS payments. If your organization wishes to use this MO, you can either use the business object (BO) supplied with the base product or configure your own.

The following points highlight some specific characteristics for this MO:

- This MO has a status, but no logs are provided.
- The standard characteristics collection is not provided.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference C1-PAY TMPL .

Fastpath: For more information, about this MO and to review the business objects defined for this MO, navigate to **Admin, Maintenance Object** and view the MO `C1-PAY TMPL`.

Payment Event Distribution

The base-package, by default, creates a single payment for a payment event. If your business requires potentially many payments to be created when payment events are added, you'll need to set up the various control tables described in this section.

Fastpath: Refer to *Distributing A Payment Event* for more information about how payment event distribution is handled in the system.

Making Payments Using Distribution Rules

As part of this method, one or more distribution details are provided at payment time along with the usual payment and tender information. Each distribution detail record references a distribution rule and a corresponding value. The distribution rule encapsulates the business rules that govern the distribution of the payment amount into payments using the specified value.

The type of value being captured on the distribution detail and the logic that uses it to create payments are defined on the *distribution rule*.

Rule Value

The primary use of the rule value is to identify the business entity whose balance is to be relieved by creating payment(s). In most cases where the payor account is the same as the payee account it may also used to identify the tender account associated with the payment(s).

Determine Tender Account

The very first step in processing a distribution detail is to identify the tender account (i.e. the payor) associated with the payment. To do that the system calls the `Determine Tender Account` *algorithm* defined on the distribution rule providing it with the rule value and other tender information.

Creating Payment(s)

The business logic that distributes a payment amount into one or more payments(s) targeted towards the entity identified by a rule value is held in designated `Create Payment` *algorithms* defined on the distribution rule.

Rule Value Can Capture Additional Information

A rule value can also be used to capture additional information provided at payment time, like address information, comments, etc. Obviously payment distribution details with this type of rule value should have a zero payment amount, as they are not real payments. These distribution details end up being linked to a payment event, but unlike other distribution details they do not contribute any payments. You can think of these details as payment event characteristics.

You don't have to set up a `Create Payment` algorithm for distribution rules intended solely to capture additional payment information.

Setting Up The System To Use Distribution Rules

Setting Up Distribution Rules

Define a Distribution Rule for each payment event distribution method practiced by your business.

Distribution Rule - Main

To set up a distribution rule, navigate to **Admin Menu, Distribution Rule**.

Description of Page

Enter a unique **Distribution Rule** and **Description** for the distribution rule.

Provide a short and unique **Distribution Rule Label** to be used as rule's name throughout the system.

Characteristic Type defines the type of entity whose balance is relieved by the payment(s) this rule creates. For example, if this rule targets payments(s) towards a specific contract, you'd reference a characteristic that its value identifies a contract. We use the term "rule value" for the characteristic value.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_DST_RULE*

Distribution Rule - Algorithms

Navigate to **Admin, Distribution Rule, Algorithms** to set up the algorithms appropriate for your distribution rule.

Description of Page

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

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to `10` unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Optional / Required	Description
`Create Payment`	Optional	This algorithm is executed to distribute a payment distribution detail payment amount into one or more payments. Click *here* to see the algorithm types available for this system event.
`Determine Tender Account`	Optional	This algorithm is executed to determine the tender account associated with the payment distribution detail. Only one such algorithm may be specified. Click *here* to see the algorithm types available for this system event.

Feature Configuration

You must set up the **Financial Transaction Options** feature configuration to define parameters that control various payment event distribution options.

The following points describe the various **Option Types** that must be defined:

- `Always Enable Distribution Rule.` This option controls whether the system should only use the distribution rule method to add payment events or rather allow both the default method and the distribution rule method to coexist.
    - Enter `Y` if the system should always use distribution rules. With this setting, navigation to the Payment Event page in add mode opens up the *Payment Event Quick Add* page (defaulting it to the `single` payment event dialog). This dialog is designed to create a payment event using distribution rules
    - Enter `N` if the system should allow both methods. With this setting, navigation to the Payment Event page in add mode opens up the standard *Payment Event - Add Dialog* that uses the default method to create a payment event. If you want to use the distribution rule method, navigate to the Payment Event Quick Add page from the menu.
- `Default Distribution Rule.` This option states your default distribution rule that appears throughout the system.

Cancel Reasons

As described in *The Financial Big Picture*, the various types of financial transactions can be canceled if their financial impact needs to be reversed from the system. Whenever a financial transaction is canceled, a cancel reason must be specified. This section describes the control tables that contain the cancel reason codes.

Setting Up Bill (Segment) Cancellation Reasons

Open **Admin Menu, Bill Cancel Reason** to define your bill segment cancellation reason codes.

Description of Page

Enter an easily recognizable **Bill Cancel Reason** and **Description** for the bill cancellation reason.

Only use **System Default** on those reason codes that are placed on bill segments that are automatically canceled by the system. Valid values are: `Turn off auto-cancel` and `Mass Cancel`. The reason code identified as `Turn off auto-cancel` is placed on bill segments that are automatically canceled when the final bill segment ends before the prior bill (and therefore we have to cancel the prior bill). The reason code identified as `Mass Cancel` is placed on bill segments that are canceled as a result of the execution of the Mass Cancellation background process. Refer to *Mass Cancellation* for more information.

Note: Required values. You must have one reason code defined for each of the System Default values.

Setting Up Payment Cancellation Reasons

Open **Admin Menu, Payment Cancel Reason** to define your payment cancellation reason codes.

Description of Page

Enter an easily recognizable **Cancel Reason** and **Description** for the payment cancellation reason.

Turn on the **NSF Charge** switch if the system should invoke the non-sufficient funds (NSF) algorithm when a tender is cancelled using this reason code. Refer to *NSF Cancellations* for more information.

The next several fields are used to change an account's credit rating or cash-only points if a tender is canceled using the respective reason code.

- Use **Affect Cash-Only Score By** to define how tenders canceled using this reason will affect the account's cash-only score. This should be a positive number. When a customer's cash-only points exceed the cash-only threshold amount defined on the CIS installation record, the account is flagged as cash only during payment processing and on Control Central.
- Use **Affect Credit Rating By** to define how tenders canceled using this reason will affect the account's credit rating. This should be a negative number. A customer's credit rating is equal to the start credit rating amount defined on the CIS installation record plus the sum of credit rating demerits that are currently in effect.
- Use **Months Affecting Credit Rating** to define the length of time the demerit remains in effect. This information is used to define the effective period of the credit rating demerit record.

Fastpath: For more information, refer to *Account - Credit Rating*.

Note: The payor gets the credit rating / cash only hit. When you cancel a tender you must specify a cancellation reason. If the cancellation reason indicates a credit rating / cash only demerit should be generated, the system levies the credit rating transaction on the PAYOR's account.

The **System Default Flag** is specified on those cancellation reasons that are placed on payment segments that are automatically cancelled by the system. Valid values are: `Re-opened Bill`. The `Re-opened Bill` value is used as follows:

- Payments are automatically created for accounts who pay their bills automatically when their bills are completed.
- If such a bill is reopened before the automatic payment is interfaced to the paying authority, the system automatically cancels the payment. The `Re-opened Bill` cancellation reason is placed on such payments.

Setting Up Adjustment Cancellation Reasons

Open **Admin Menu, Adjustment Cancel Reason** to define your adjustment cancellation reason codes.

Description of Page

Enter an easily recognizable **Cancel Reason** and **Description** for each adjustment cancellation reason.

Miscellaneous Financial Controls

This section describes miscellaneous control tables.

A/P Check Request

Adjustments whose adjustment type is marked with an A/P check request code are interfaced to your A/P system. Your A/P system then cuts the checks.

Fastpath: Refer to *Controls The Interface To A/P* for more information about the accounts payable interface.

You must set up at least one A/P check request code if you want A/P to cut checks.

To set up A/P check request types, open **Admin Menu, A/P Request Type**.

Description of Page

Enter an easily recognizable **A/P Request Type** for the accounts payable request type.

Use **Due Days** to define when the check is cut. The cut date is equal to the adjustment date plus due days.

Select a **Pay Method**. Choose from these options:

`System Check` System check

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_APREQ_TYPE*.

Billable Charge Line Type

Oracle Revenue Management and Billing enables you to create a charge line in a pass through billable charge using a billable charge line type. It is a billable charge line type which helps the system to determine the following:

- The description which should appear on the bill for a charge line which is created using the billable charge line type.
- The currency in which the charge line should be created.
- Whether the charge line should appear on the person’s printed bill.
- Whether the charge line should be included in the bill summary.
- Whether the charge line should be considered while calculating the bill amount.
- The distribution code which indicates the GL account to which the charge line should be posted.

You can also define characteristics for a billable charge line type. The **Pricing Rule Type Category (C1PRTCAT)** characteristic type is shipped with the product. You can define the pricing rule type category characteristic for a billable charge line type. It helps to associate a billable charge line type with one or more pricing rule type categories. You can then use the billable charge line type while defining pricing rule types or pricing rules for the respective category in the self-funded health care business. Note that, at present, you can only use this feature while defining the ancillary pricing rules. For example, if the BCLT1 and BCLT2 billable charge line types are associated with the **Ancillary** pricing rule type category, then you can use BCLT1 and/or BCLT2 while defining the ancillary pricing rules. In other words, you will not be able to use a billable charge line type while defining an ancillary pricing rule until the billable charge line type is associated with the **Ancillary** pricing rule type category.

The **Billable Charge Line Type** screen allows you to define, edit, copy, and delete a billable charge line type. It contains the following zones:

- *Billable Charge Line Type List* on page 206 zone

- *Billable Charge Line Type* on page 206 zone

Billable Charge Line Type List

The **Billable Charge Line Type List** zone lists the billable charge line types which are already defined in the system. It contains the following columns:

Column Name	Column Description
Billable Charge Line Type	Displays the billable charge line type.
Description	Displays the description of the billable charge line type.
Edit	On clicking the **Edit** ( ) icon, the **Billable Charge Line Type** screen appears where you can edit the details of the billable charge line type.
Duplicate	On clicking the **Duplicate** ( ) icon, the **Billable Charge Line Type** screen appears where you can define a new billable charge line type using an existing billable charge line type.
Delete	On clicking the **Delete** ( ) icon, you can delete the billable charge line type. **Note:** You can delete a billable charge line type only when it is not yet used in the system.

On clicking the **Broadcast** ( ) icon corresponding to a billable charge line type, the **Billable Charge Line Type** zone appears with the details of the respective billable charge line type.

Related Topics

For more information on…	See...
How to edit a billable charge line type	*Editing a Billable Charge Line Type* on page 212
How to copy a billable charge line type	*Copying a Billable Charge Line Type* on page 210
How to delete a billable charge line type	*Deleting a Billable Charge Line Type* on page 213
How to view the details of a billable charge line type	*Viewing the Billable Charge Line Type Details* on page 214

Billable Charge Line Type

The **Billable Charge Line Type** zone displays the details of the billable charge line type. It contains the following sections:

- **Main** - Displays basic information about the billable charge line type. It contains the following fields:

Field Name	Field Description
Billable Charge Line Type	Displays the billable charge line type.
Description on Bill	Displays the description which should appear on the bill for a charge line which is created using the billable charge line type.
Currency	Indicates the currency in which you want to create the charge line.
Show on Bill	Indicates whether you want the charge line to appear on the person’s printed bill.
Appears in Summary	Indicates whether you want to include the charge line in the bill summary.

Field Name	Field Description
Memo Only, No GL	Indicates whether you want to include the charge line while calculating the bill amount.
Distribution Code	Displays the distribution code which indicates the GL account where you want to post the charge line. **Note:** It has a link. On clicking the link, the **Distribution Code** screen appears where you can view the details of the respective distribution code.

- **Characteristics** - Lists the characteristics defined for the billable charge line type. It contains the following columns:

Column Name	Column Description
Sequence	Used to indicate the sequence number of the characteristic.
Characteristic Type	Displays the characteristic type.
Characteristic Value	Displays the value of the characteristic type.

- **Record Actions** - This section contains the following buttons:

Button Name	Button Description
Edit	Used to edit the details of the billable charge line type.
Delete	Used to delete the billable charge line type. **Note:** You can delete a billable charge line type only when it is not yet used in the system.
Duplicate	Used to create a new billable charge line type using an existing billable charge line type.

- **Record Information** - This section contains the following field:

Field Name	Field Description
Business Object	Indicates the business object using which the billable charge line type is created. In addition, a context menu appears corresponding to this field which helps in navigating to other screens in the application. **Note:** It has a link. On clicking the link, the **Business Object** screen appears where you can view the details of the respective business object.

By default, the **Billable Charge Line Type** zone does not appear in the **Billable Charge Line Type** screen. It appears only when you click the **Broadcast** () icon corresponding to a billable charge line type in the **Billable Charge Line Type List** zone.

Related Topics

For more information on…	See...
How to edit a billable charge line type	*Editing a Billable Charge Line Type* on page 212
How to delete a billable charge line type	*Deleting a Billable Charge Line Type* on page 213
How to copy a billable charge line type	*Copying a Billable Charge Line Type* on page 210

For more information on…	See...
How to view the details of a billable charge line type	*Viewing the Billable Charge Line Type Details* on page 214

Defining a Billable Charge Line Type

Prerequisites

To define a billable charge line type, you should have:

- Currencies and distribution codes defined in the application

Procedure

To define a billable charge line type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **B** and then click **Billable Charge Line Type**.

   A sub-menu appears.

3. Click the **Add** option from the **Billable Charge Line Type** sub-menu.

   The **Billable Charge Line Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the billable charge line type.
   - **Characteristics** - Used to define characteristics for the billable charge line type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Billable Charge Line Type	Used to specify the billable charge line type.	Yes
Business Object	Indicates the business object using which you are defining the billable charge line type.	Not applicable
Description on Bill	Used to specify the description which should appear on the bill for a charge line which is created using the billable charge line type.	Yes
Currency	Used to indicate the currency in which you want to create the charge line.	Yes
Show on Bill	Used to indicate whether you want the charge line to appear on the person's printed bill.	No
Appears in Summary	Used to indicate whether you want to include the charge line in the bill summary.	No
Memo Only, No GL	Used to indicate whether you want to include the charge line while calculating the bill amount.	No

Field Name	Field Description	Mandatory (Yes or No)
Distribution Code	Used to specify the distribution code which indicates the GL account where you want to post the charge line. **Note:** This field is disabled when the **Memo Only, No GL** option is selected. The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the **Distribution Code Search** window appears.	Yes (Conditional) **Note:** This field is required when the **Memo Only, No GL** option is not selected.

Tip: Alternatively, you can access this screen by clicking the **Add** button in the **Page Title** area of the **Billable Charge Line Type** screen.

4. Enter the required details in the **Main** section.
5. Define characteristics for the billable charge line type, if required.
6. Click **Save**.

   The billable charge line type is defined.

Related Topics

For more information on...	See...
Billable Charge Line Type screen	*Billable Charge Line Type* on page 205
How to define characteristics for a billable charge line type	*Defining Characteristics for a Billable Charge Line Type* on page 209

Defining Characteristics for a Billable Charge Line Type

Prerequisites

To define characteristics for a billable charge line type, you should have:

- Characteristic types defined in the application (where the characteristic entity is set to **Billable Charge Line Type**).

Procedure

To define characteristics for a billable charge line type:

1. Ensure that the **Characteristics** section is expanded when you are defining, editing, or copying a billable charge line type.

   The **Characteristics** section contains the following fields in a grid:

Field Name	Field Description	Mandatory (Yes or No)
Sequence	Used to specify the sequence number of the characteristic.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the billable charge line type.

Field Name	Field Description	Mandatory (Yes or No)
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types where the characteristic entity is set to **Billable Charge Line Type**.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the billable charge line type.
Characteristic Value	Used to specify the value for the characteristic type. **Note:** On specifying the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field.	Yes (Conditional) **Note:** This field is required when you are defining a characteristic for the billable charge line type.

2. Enter the required details in the **Characteristics** section.

Note: If you select a predefined characteristic type, the **Search** ( ) icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, you can search for a predefined characteristic value.

3. If you want to define more than one characteristic for the billable charge line type, click the **Add** ( ) icon and then repeat step 2.

Note: However, if you want to remove a characteristic from the billable charge line type, click the **Delete** ( ) icon corresponding to the characteristic.

Related Topics

For more information on...	See...
How to define a billable charge line type	*Defining a Billable Charge Line Type* on page 208
How to edit a billable charge line type	*Editing a Billable Charge Line Type* on page 212
How to copy a billable charge line type	*Copying a Billable Charge Line Type* on page 210

Copying a Billable Charge Line Type

Instead of creating a billable charge line type from scratch, you can create a new billable charge line type using an existing billable charge line type. This is possible through copying a billable charge line type. On copying a billable charge line type, the details including the characteristics are copied to the new billable charge line type. You can then edit the details, if required.

Prerequisites

To copy a billable charge line type, you should have:

- Billable charge line type (whose copy you want to create) defined in the application
- Currencies and distribution codes defined in the application

Procedure

To copy a billable charge line type:

1. Click the **Admin** link in the **Application** toolbar.

A list appears.

2. From the **Admin** menu, select **B** and then click **Billable Charge Line Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Billable Charge Line Type** sub-menu.

   The **Billable Charge Line Type** screen appears.

4. In the **Billable Charge Line Type List** zone, click the **Duplicate** () icon in the **Duplicate** column corresponding to the billable charge line type whose copy you want to create.

   The **Billable Charge Line Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the billable charge line type.
   - **Characteristics** - Used to define characteristics for the billable charge line type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Billable Charge Line Type	Used to specify the billable charge line type.	Yes
Business Object	Indicates the business object using which you are defining the billable charge line type.	Not applicable
Description on Bill	Used to specify the description which should appear on the bill for a charge line which is created using the billable charge line type.	Yes
Currency	Used to indicate the currency in which you want to create the charge line.	Yes
Show on Bill	Used to indicate whether you want the charge line to appear on the person's printed bill.	No
Appears in Summary	Used to indicate whether you want to include the charge line in the bill summary.	No
Memo Only, No GL	Used to indicate whether you want to include the charge line while calculating the bill amount.	No
Distribution Code	Used to specify the distribution code which indicates the GL account where you want to post the charge line. **Note:** This field is disabled when the **Memo Only, No GL** option is selected. The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the **Distribution Code Search** window appears.	Yes (Conditional) **Note:** This field is required when the **Memo Only, No GL** option is not selected.

Tip: Alternatively, you can copy a billable charge line type by clicking the **Duplicate** button in the **Billable Charge Line Type** zone.

5. Enter the required details in the **Main** section.
6. Define, edit, or remove characteristics from the billable charge line type, if required.

7. Click **Save**.

   The new billable charge line type is defined.

Related Topics

For more information on...	See...
Billable Charge Line Type screen	*Billable Charge Line Type* on page 205
Billable Charge Line Type List zone	*Billable Charge Line Type List* on page 206
Billable Charge Line Type zone	*Copying a Billable Charge Line Type* on page 210
How to define characteristics for a billable charge line type	*Defining Characteristics for a Billable Charge Line Type* on page 209

Editing a Billable Charge Line Type

Prerequisites

To edit a billable charge line type, you should have:

- Currencies and distribution codes defined in the application

Procedure

To edit a billable charge line type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **B** and then click **Billable Charge Line Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Billable Charge Line Type** sub-menu.

   The **Billable Charge Line Type** screen appears.

4. In the **Billable Charge Line Type List** zone, click the **Edit** (✎) icon in the **Edit** column corresponding to the billable charge line type whose details you want to edit.

   The **Billable Charge Line Type** screen appears. It contains the following sections:

   - **Main** - Used to specify basic details about the billable charge line type.
   - **Characteristics** - Used to define characteristics for the billable charge line type.

   The **Main** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Billable Charge Line Type	Displays the billable charge line type.	Not applicable
Business Object	Indicates the business object using which you are defining the billable charge line type.	Not applicable
Description on Bill	Used to specify the description which should appear on the bill for a charge line which is created using the billable charge line type.	Yes
Currency	Used to indicate the currency in which you want to create the charge line.	Yes

Field Name	Field Description	Mandatory (Yes or No)
Show on Bill	Used to indicate whether you want the charge line to appear on the person's printed bill.	No
Appears in Summary	Used to indicate whether you want to include the charge line in the bill summary.	No
Memo Only, No GL	Used to indicate whether you want to include the charge line while calculating the bill amount.	No
Distribution Code	Used to specify the distribution code which indicates the GL account where you want to post the charge line. **Note:** This field is disabled when the **Memo Only, No GL** option is selected. The **Search** ( ) icon appears corresponding to this field. On clicking the **Search** icon, the **Distribution Code Search** window appears.	Yes (Conditional) **Note:** This field is required when the **Memo Only, No GL** option is not selected.

Tip: Alternatively, you can edit the details of a billable charge line type by clicking the **Edit** button in the **Billable Charge Line Type** zone.

5. Modify the required details in the **Main** section.
6. Define, edit, or remove characteristics from the billable charge line type, if required.
7. Click **Save**.

   The changes made to the billable charge line type are saved.

Related Topics

For more information on...	See...
Billable Charge Line Type screen	*Billable Charge Line Type* on page 205
Billable Charge Line Type List zone	*Billable Charge Line Type List* on page 206
Billable Charge Line Type zone	*Billable Charge Line Type* on page 206
How to define characteristics for a billable charge line type	*Defining Characteristics for a Billable Charge Line Type* on page 209

Deleting a Billable Charge Line Type

Procedure

To delete a billable charge line type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.
2. From the **Admin** menu, select **B** and then click **Billable Charge Line Type**.

   A sub-menu appears.
3. Click the **Search** option from the **Billable Charge Line Type** sub-menu.

   The **Billable Charge Line Type** screen appears.

4. In the **Billable Charge Line Type List** zone, click the **Delete** ( ) icon in the **Delete** column corresponding to the billable charge line type that you want to delete.

   A message appears confirming whether you want to delete the billable charge line type.

   Note: You can delete a billable charge line type only when it is not yet used in the system.

   Tip: Alternatively, you can delete a billable charge line type by clicking the **Delete** button in the **Billable Charge Line Type** zone.

5. Click **OK**.

   The billable charge line type is deleted.

Related Topics

For more information on...	See...
Billable Charge Line Type screen	*Billable Charge Line Type* on page 205
Billable Charge Line Type List zone	*Billable Charge Line Type List* on page 206
Billable Charge Line Type zone	*Billable Charge Line Type* on page 206

Viewing the Billable Charge Line Type Details

Procedure

To view the details of a billable charge line type:

1. Click the **Admin** link in the **Application** toolbar.

   A list appears.

2. From the **Admin** menu, select **B** and then click **Billable Charge Line Type**.

   A sub-menu appears.

3. Click the **Search** option from the **Billable Charge Line Type** sub-menu.

   The **Billable Charge Line Type** screen appears.

4. In the**Billable Charge Line Type List** zone, click the **Broadcast** ( ) icon corresponding *t*o the billable charge line type whose details you want to view.

   The **Billable Charge Line Type** zone appears.

5. View the details of the billable charge line type in the **Billable Charge Line Type** zone.

Related Topics

For more information on…	See…
Billable Charge Line Type screen	*Billable Charge Line Type* on page 205
Billable Charge Line Type List zone	*Billable Charge Line Type List* on page 206
Billable Charge Line Type zone	*Billable Charge Line Type* on page 206

Payables Cash Accounting

In some areas, taxes and other 3 rd party liabilities are not truly payable until the customer remits payment. We refer to this as "payables *cash* accounting". This practice should be contrasted with "payables *accrual* accounting" in which the liability is realized when the bill is created (as opposed to when it is paid).

Note: Value Add Tax (VAT). VAT is a form of taxation common throughout the European Union. It is a common practice to only book the VAT payable when the customer remits payment. This means that most European implementations will use the functionality described in this section.

If your organization does not practice payable cash accounting, you may skip this section as accrual accounting is the system default. If you practice payables cash accounting, the contents of this section describe how to configure the system appropriately.

Accrual versus Cash Accounting Example

The following is an example of the financial events that transpire when a customer is billed and payment is received using *accrual* accounting.

Event	GL Accounting	Tax Payable Balance
Bill segment created	A/R 110 Revenue <100> Tax Payable <10>	(10)
Payment received	Cash 110 A/R <110>	(10)

In the above example, you'll notice that the payable is booked when the bill is created. Let's contrast this with what takes place if the payable is subject to payables *cash* accounting.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	0	(10)
Payment received	Cash 110 A/R <110> Tax Holding 10 Tax Payable <10>	(10)	0

Notice that when the bill segment is produced, the liability is not booked, rather, the amount of the liability is placed in a "holding" GL account. When the customer pays, the moneys are transferred from the "holding" GL account to the true tax payable account.

Note: Cash accounting is only applicable for liabilities. In the above example, you'll notice that only the tax payable account had cash accounting implications. This is because organizations that practice cash accounting only do it for liability accounts; they never do it for assets, revenue or expenses.

If the above seems simple, consider the following complications that must be considered:

- What happens if a partial payment is received?

- What happens if there are multiple taxes subject to cash accounting rules?
- What happens if the A/R is relieved via a deposit seizure (or transference of a credit balance from another contract)?
- What if, after payment, the original bill segment is cancel/rebilled resulting in a different amount of tax (keep in mind that the payable got booked when the payment was received)?
- What happens if the payment is cancelled?
- What if the payment isn't received and we have to write-off debt?
- What happens if the customer overpays?
- What happens if the customer is allowed to prepay their tax (this is a common practice in the United Kingdom) and then the tax rate changes at billing time?

The above points, and more, are discussed below.

Distribution Code Controls Cash Accounting For A GL Account

Note: If you do not understand the significance of distribution codes, please refer to *Setting Up Distribution Codes*.

Whether or not cash accounting is used for a specific GL account is defined on HOLDING GL account's distribution code (i.e., the holding GL account references the true payable account).

It is very important that unique payable and holding distribution codes be used for each type of tax subject to cash accounting rules. For example, if you have cash accounting requirements for both value-added tax (VAT) and a climate levy, you would need four distribution codes:

- VAT Payable.
- VAT Holding.
- Climate Levy Payable.
- Climate Levy Holding.

Without unique distribution codes for each payable and holding account, the system cannot keep track of how much of a given tax is being held, awaiting payment.

Bill Segments and Cash Accounting

The contents of this section describe how cash accounting is implemented when bill segments are created.

Rate Component Distribution Codes

The distribution codes on rate components that calculate tax must be your HOLDING payable distribution codes.

Bill Segment Financial Transactions Are Not Affected By Cash Accounting

There are NO changes to rate calculation associated with cash accounting. This is because the rate components that calculate tax reference the HOLDING payable distribution codes.

Note: Prepaid taxes - future functionality. If your organization allows customers to prepay taxes in anticipation of a future tax increase (the customers receive the lower rate if they pay in advance), please speak to your account manager for information about when corresponding functionality will be available.

Payment Segments and Cash Accounting

The contents of this section describe how cash accounting is implemented when payment segments are created.

Payment Segment Financial Transaction Algorithms Transfer Holding Amounts to Payable GL Accounts

Logic exists in the pay segment's FT algorithm that transfers amounts from payable holding distribution codes to their respective payable real distribution codes.

Fastpath: Refer to *Setting Up Payment Segment Types* for how to define the appropriate FT algorithm.

The following table shows what happens to the financial transaction associated with the payment segment for a cash accounting customer.

Event	GL Accounting
Bill segment is created	A/R 110 Revenue <100> Tax Holding <10>
Payment segment relieves receivables	Cash 110 A/R <110>
Additional GL details created when the payment segment FT algorithm transfers the holding amount to a payable account	Tax Holding 10 Tax Payable <10>
Net affect of the above	Cash 110 A/R <110> Tax Holding 10 Tax Payable <10>

How Does The System Know What Amounts To Transfer From Holding To Payables?

When a payment segment is created for an account that is subject to cash accounting processing, the system determines if there is a CREDIT balance for any *holding* distribution code in respect of the contract. If so, it generates additional GL details to transfer moneys from the holding distribution code to the payable distribution code in proportion to the amount of receivables relieved by the payment. Therefore, if 100% of receivables are relieved by the payment segment, 100% of the holding amounts will be transferred to payable distribution codes. Refer to *Partial Payments Result In Partial Payables* for an example of what happens when a partial payment is created.

Partial Payments Result In Partial Payables

The previous example showed the entire tax holding amount being transferred to the tax payable account. The entire holding amount was transferred because the contract was paid in full. If a partial payment is received, only part of the holding amount will be transferred to the payable amount (proportional to the amount of receivables reduced by the payment). An example will help make the point.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	0	(10)

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Partial payment received	Cash 27.50 A/R <27.50> Tax Holding 2.50 Tax Payable <2.50>	(2.50)	(7.50)

Adjustments That Behave Like Payments

There are several types of adjustments that behave just like payments (in respect of payables cash accounting). Consider the following events:

- Seizing a deposit (i.e., transferring a credit from a deposit contract to a regular contract)
- Overpayments transferred from one contract to another

The above events should cause the system to transfer holding amounts to true payable amounts (notice that the above examples are all transfer adjustments).

However, there are many other adjustments that should NOT behave like payments. You control how the adjustment works by selecting the appropriate FT algorithm when you *set up adjustment types* (refer to *ADJT-AC* and *ADJT-TC* for a description of the base package algorithms that causes the holding amounts to be manipulated in proportion to the amount of receivable being adjusted). In other words, there are adjustment FT algorithms that cause the transference of holding payable amounts to real payable amounts when the A/R balance is decreased by the adjustment.

Note: Cash refunds can behave like "anti-payments". In addition to the above examples of transfer adjustments behaving like payments, you should be aware that cash refunds may impact your holding and true payable balances. Refer to *Cash Refunds* for more information.

Overpayment Of Taxes Due To Cancel/Rebills

Lets assume a cancel / rebill occurs after a payment is received and the net affect of the cancel / rebill is that the customer has overpaid their taxes.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	0	(10)
Payment received	Cash 110 A/R <110> Tax Holding 10 Tax Payable <10>	(10)	0
Cancel	A/R <110> Revenue 100 Tax Holding 10	(10)	10
Rebill	A/R 27.50 Revenue <25> Tax Holding <2.50>	(10)	7.50

You'll notice that the amount payable to the taxing authority still indicates $10 (the amount of tax that was paid by the customer). However, you'll notice that the tax holding balance is 7.50 (debit). This looks a bit odd, but it's correct. Remember that at this point, the customer has a credit balance of $75 and this will be whittled down as successive bills are produced as shown below. Note: refer to *Cash Refunds* for an example of what happens if you refund the credit with a check rather than letting it whittle down.

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance
		(10)	7.50
Bill segment created	A/R 55 Revenue <50> Tax Holding <5>	(10)	2.50
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	(10)	(7.50)

In the unlikely event of a payment being received while the tax holding has a debit balance, nothing will be done in respect of transferring funds from holding to payable (there is nothing to transfer).

Cash Refunds

If you refund moneys to a cash accounting customer, it's important to do the opposite of what was done when the payment was received (i.e., you need to transfer the payable back to the holding account). The following example should help clarify this situation (this example shows a refund due to a credit balance that occurred as a result of a cancel/rebill).

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance	Contract's Payoff Balance
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	0	(10)	110
Payment received	Cash 110 A/R <110> Tax Holding 10 Tax Payable <10>	(10)	0	0
Cancel	A/R <110> Revenue 100 Tax Holding 10	(10)	10	(110)
Rebill	A/R 27.50 Revenue <25> Tax Holding <2.50>	(10)	7.50	(82.50)

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance	Contract's Payoff Balance
Payment refunded (via an A/P adjustment)	Cash <82.50> A/R 82.50 Tax Holding <7.50> Tax Payable 7.50	(2.50)	0	0

We understand this is tricky, but consider this - when a cash accounting customer makes a payment, the system transfers tax holding CREDIT balances to tax payable distribution codes in proportion to the amount of the receivable DEBIT amount that was reduced by the payment. Therefore, when cash is returned to the customer, the system should transfer tax holding DEBIT balances to tax payable distribution codes in proportion to the amount of the receivable CREDIT that was reduced by the refund.

Note: The above takes place when an A/P adjustment is created if the related adjustment type references the appropriate FT algorithm (refer to *adjustment FT algorithm used for adjustments that behave like payments*.

Over Payments

If a customer overpays a bill (i.e., we receive more cash than receivables), we strongly recommend you set up the system to NOT keep the excess credit on the customer's regular contracts. Rather, we recommend you segregate the receivable onto an "excess credit" contract. If you do this, the system will transfer any excess credits to the regular contracts at bill completion time. When this transfer occurs, the same accounting described under *Payments Segment Financial Transaction Algorithms Transfer Holding Amounts To Payable GL Accounts* occurs as shown in the following example. Note: this example assumes an excess credit of $110 was transferred to a normal contract and the normal contract had $10 of held payables.

Fastpath: Refer to *Overpayment Segmentation* for how to set up the system to segregate overpayments on a separate contract.

Note: Why not keep excess credits on a customer's regular contract? Because the system can't differentiate between a credit that exists as a result of an overpayment and a credit that exists because of cancel/rebills, it would be impossible for the system to know if payables should be realized as a result of the reduced credit balance. However, if you keep overpayments on an excess credit contract, the system knows to treat any transference of these credits as "payments" and therefore it can transfer holding balances to true payables.

Event	Normal Contract GL Accounting	Excess Credit Contract GL Accounting
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	
Payment of $300 is received	Cash 110 A/R <110> Tax Holding 10 Tax Payable <10>	Cash 190 Overpay <190>

Event	Normal Contract GL Accounting	Excess Credit Contract GL Accounting
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	
Transfer excess credit amount to normal contract (when bill is completed).	Xfer 110 A/R <110>	Overpay 110 Xfer <110>
Because the transfer adjustment is the equivalent of a cash relief outstanding tax holding is relieved in proportion to the amount of receivables that are reduced by the transfer	Tax Holding 10 Tax Payable <10>	
Net affect of the transfer	Xfer 110 A/R <110> Tax Holding 10 Tax Payable <10>	Overpay 110 Xfer <110>

Note: Prepaid taxes - future functionality. If your organization allows customers to prepay taxes in anticipation of a future tax increase (the customers receive the lower rate if they pay in advance), we do *not* consider this prepayment to be an overpayment. Rather, it is a payment of future taxes that will be remitted to the taxing authorities at payment time (due to cash accounting). Please speak to your account manager for when corresponding functionality will be available.

Write Down Adjustment

Writing down debt is very different from *writing off debt*. When you write down debt, you are removing the receivable with no expectation of it being paid. For example, most organizations write down small debit and credit balances as part of their write-off process (e.g., they don't send a very small amount to a collection agency).

Let's run through an example to illustrate this:

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance	Contract's Payoff Balance
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>	0	(10)	110
Payment received	Cash 109.50 A/R <109.50> Tax Holding 9.95 Tax Payable <9.95>	(9.95)	.05	0.50

Event	GL Accounting	Tax Payable Balance	Tax Holding Balance	Contract's Payoff Balance
Write down cash accounting debt	Tax Holding 0.05 Write Down Expense 0.45 A/R <0.50>	(9.95)	0	0

In order to achieve the above, you must set up an adjustment type that references a special financial transaction algorithm (refer to *cash accounting write down algorithm* for more information). This algorithm will reduce / increase the receivable balance accordingly AND cause any holding amounts to be set to zero. This adjustment type should be referenced on your write-down algorithm that is referenced on your write-off controls.

Write-Offs

At write-off time we may refund credit balances. The refunding of credit balances is handled by A/P adjustments and these have cash accounting processing as described under *Cash Refunds*.

If we have to write-off debt, holding balances are relieved in proportion to the amount of debt that is written off (as usual). It's important to understand that for this to work, you must set up the system as follows:

- The tax holding distribution codes must have their override distribution switch turned on.
- The distribution code on the contract type associated with the contract to which the written-off payables are transferred must be the REAL payable distribution codes. This is important so that if the customer pays after the payables are reversed, we will be able to debit cash and credit the REAL payable distribution code.

Let's run through an example to illustrate this.

Event	Normal Contract GL Accounting	Write Off Revenue Contract GL Accounting	Reverse Liabilities Contract GL Accounting
Bill segment created	A/R 110 Revenue <100> Tax Holding <10>		
Write Off Time			
Reverse the held payables	Xfer 10 A/R <10>		Tax Holding 10 Xfer <10> Note, the tax holding only gets debited if you have turned on the override at write-off switch on its distribution code
Write off revenue	Xfer 100 A/R <100>	Write Off Expense 100 Xfer <100>	

Event	Normal Contract GL Accounting	Write Off Revenue Contract GL Accounting	Reverse Liabilities Contract GL Accounting
If the customer subsequently pays		Cash 100 Write Off Exp <100>	Cash 10 Tax Payable <10> Note, the tax payable only gets credited if the contract type's distribution code has been defined as such

Open Item Accounting

The topics in this section provide background information about open-item accounting.

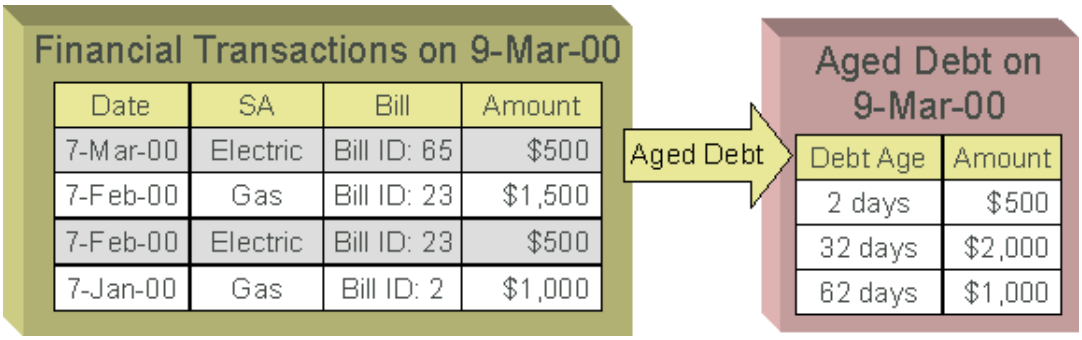
Note: This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization practices open-item accounting. If your organization practices balance-forward accounting, you need only indicate such on your *customer classes*; no other setup is required. Refer to *Open Item Versus Balance Forward Accounting* for more information about these two accounting practices.

Open-Item Versus Balance-Forward Accounting

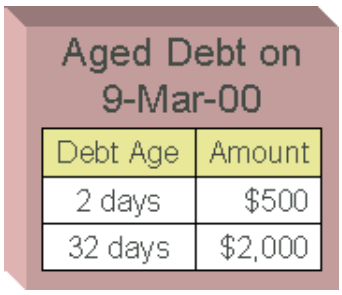
If you practice open-item accounting, you match payments against bills. The term "open-item accounting" is used to describe this accounting practice because:

- Payments are matched against "open items" (i.e., unpaid bills and adjustments)
- Only unmatched bills and adjustments (i.e., open items) affect aged debt.

Contrast open-item accounting with "balance-forward" accounting - in a balance-forward world, payments are not matched to bills. Rather, payments implicitly relieve a customer's oldest debt. For example, consider the following unpaid financial transactions that exist for an account and the resultant aged debt.

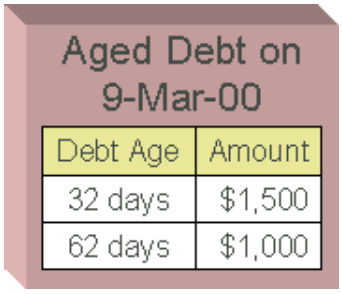


In a balance-forward world, if a $1,000 payment was made on 9-Mar-00, the customer's aged debt would look as follows:



Notice how the $1,000 payment relieves the 62 day old debt - it does this because, in a balance-forward world, payments payoff oldest debt first.

However, let's assume the customer wants the payment to settle his electric debt (e.g., because he disagrees with the gas bills). If you could match the $1,000 payment to the two electric bills (i.e., open-item accounting exists), the customer's aged debt would look as follows:



In sum,

- In an open-item world, payments are matched to bills and only unpaid bills and adjustments (i.e., open items) affect aged debt.
- In a balance-forward world, payments are not matched to bills and therefore a customer's aged debt is computed by aging debits (bills and adjustments) and then relieving the oldest debits using credits (payments and adjustments).

Note: Financial Transactions and Bills. In an open-item world, only bill segments and adjustments are presented on a bill. When a bill is completed, only those bill segments and adjustments to be presented are swept onto a bill. Payment and payment cancellation FTs, bill segment FTs canceled before bill completion together with their corresponding bill segment cancellation FTs, and adjustment FTs marked as do not show on bill are not swept onto a bill. An adjustment's adjustment type and its algorithms determine if its FT will show on a bill by default.

Accounting Method Defined On Your Customer Classes

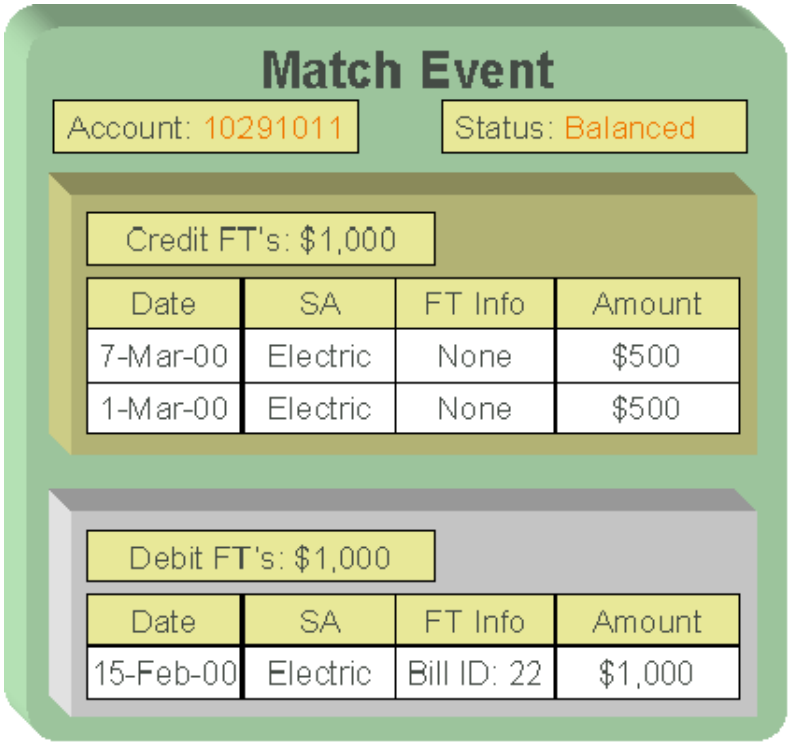
You define the type of accounting method that is practiced (*balance-forward versus open-item*) on your *customer classes*. For example, residential customers can practice balance-forward accounting whereas industrial / commercial customers can practice open-item accounting.

Match Events

Match events are used to match open-items (i.e., debit and credit financial transactions) together. The topics in this section provide an overview of match events.

Match Events Match Debit FTs To Credit FTs

For open-item customers, the system matches credit financial transactions (FT's) to debit FT's under a *match event*. The following is an example of a match event associated with two $500 payments that satisfy the debt associated with one bill (on February 2000).



Notice the following:

- The match event matches 2 credit FT's against a single debit FT. A match event may contain an unlimited number of FT's.
- The match event contains FT's associated with a single account. While the FT's under a match event may belong to multiple contracts, all FT's under a match event must belong to the same account.
- The match event only contains bill segments that belong to a single bill. If you mix multiple bills under a single match event, then an individual bill balance cannot be properly determined when partial payments exist.
- The status of the match event is `balanced`. This is because the sum of the debits equals the sum of the credits. If debits do not equal credits, the status of the match event would be `open` and the various FT's would still affect the customer's aged debt. Refer to *Match Event Lifecycle* for more information.

Caution: Warning: It is strongly encouraged that you refrain from mixing multiple bills on a single match event. If you stick by the rule of "just one bill per match event" you will then be able to determine the outstanding balance of a partially paid bill (see the *bill page*, bill summary section). However, if you mix more than one bill under a match event, a particular bill's balance may become indeterminate. Algorithm types have been provided which help to enforce this rule of "one bill per match event", please refer to *Match By Bill, Pay Oldest Bill First* for an example of a matching algorithm that enforces this notion.

When Are Match Events Created?

The following points describe when match events are created for open-item accounts:

Note: Match events are only created for open-item accounts (i.e., those accounts with a customer class that indicates open-item accounting is practiced). Match events may not be created for balance-forward accounts.

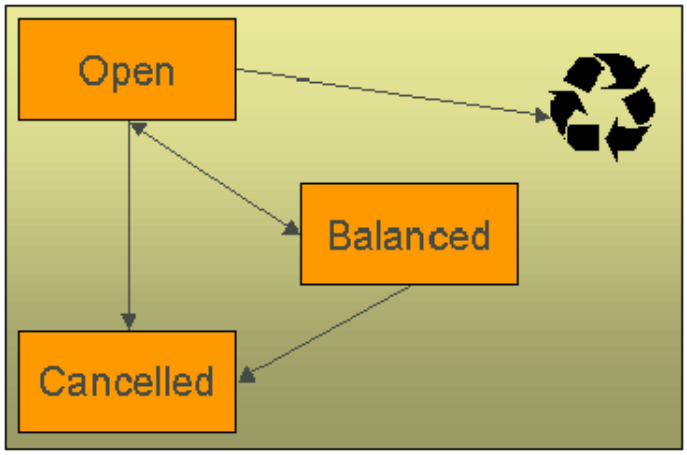
- The system can create one or many match events when a payment is added. This match event matches the payment's credit FT's with the debit and credit FT's from bill segments and adjustments. The FT's that are linked to the match event are controlled by the payment's **match type** and **match value** (payments made by open-item customers must reference a match type and match value). Refer to *Payments And Match Events* for more information.
- The system may create a match event when any type of financial transaction is cancelled. This match event groups together the original FT with its cancellation FT. Refer to *How Are Match Events Cancelled?* for more information.
- The system creates a match event when a bill is completed for customers that pay automatically (i.e., direct debit customers). The match event groups together the bill's new charges against the automatic payment's payment segments.
- The system creates a match event when a bill is completed where the new charges are offset by other financial transactions. For example:
  - Consider a bill that contains a deposit refund. If the sum of the deposit refund equals or exceeds the amount of the bill, the bill's FT's can be matched against the debit refunds FT's.
  - Consider a bill whose new charges are offset by a previous overpayment. Refer to *Over Payments* for more information about overpayments.

Fastpath: Refer to *Bill Lifecycle* for more information about what happens during bill completion.

- The system creates a match event when a contract closes and the contract has unmatched FT's. For example:
  - Consider a deposit contract that closes when the deposit is refunded to the customer. The system will create a match event with the deposit contract's FT's (the original credit and the debits used to refund the deposit) when the deposit contract closes (i.e., when its credit balance falls to zero).
  - Consider a contract for utility debt that is written off. This contract closes when the system creates transfer adjustments to transfer the utility debt to a write-off contract (or writes down the debt). The system creates a match event to match the original debt to the transfer adjustments used to write-off the debt. Refer to *How Is Debt Financially Written Off* for more information about write-off processing.
- A user can create a match event manually at any time. Manual match events would be created under a variety of situations. For example:
  - If a customer disputes a charge. Refer to *Disputing Items* for more information about disputes.
  - To handle unusual situations when the system is unable to automatically match FT's together.

Match Event Lifecycle

The following diagram shows the possible lifecycle of a match event:



Match events are initially created in the `open` state. Financial transactions (FT's) linked to `open` match events affect arrears, but not in an open-item fashion. Rather, FT's linked to `open` match events affect arrears in a balance-forward fashion. Refer to *Open Item Versus Balance Forward Accounting* for more information about these two accounting methods.

A user may delete an `open` match event. When an `open` match event is deleted, its FT's may be linked to other match events.

The system automatically changes an `open` event's status to `balanced` when the sum of the debit financial transactions (FT's) equals the sum of the credit FT's *for each contract on the match event*. It's worth stressing that a match event may contain FT's from many contract's and each contract's FT's must sum to zero before the match event can become `balanced`.

A user may re `open` a `balanced` event (by adding / removing FT's so that the match event becomes unbalanced).

A user may `cancel` a `balanced` or `open` match event. Refer to *How Are Match Events Cancelled?* for more information about cancellation.

Payments And Match Events

As described under *When Are Match Events Created?*, the system creates a match event when a payment is added for an open-item account. The system uses the payment's **match type** and **match value** to determine the FT's (e.g., bill segments and adjustments) that will be matched with the payment's FT's (i.e., the payment segments).

Another way to think of this is as follows:

- When most payments are distributed, the system calls the payment distribution algorithm that is plugged-in on the account's customer class.
- However, a payment that is made in respect of a specific bill requires a different distribution algorithm because the payment should only be distributed amongst the debt associated with the specific bill being paid. This is accomplished by referencing a match type / match value on the payment. The match type references the appropriate payment distribution algorithm. This algorithm is used rather than the customer class distribution algorithm.

For example, if a payment were made in respect of bill ID `192910192101`, this payment would reference a match type of `bill ID` and a match value of `192910192101`. At payment distribution time, the system calls the override payment distribution algorithm associated with this match type. The base package bill ID distribution algorithm does several things:

- It distributes the payment amongst contracts associated with the bill.
- It creates a match event and links the bill's bill segment and adjustment FT's to it.
- Refer to the *Bill ID Match Type Algorithm* for more information about this algorithm.

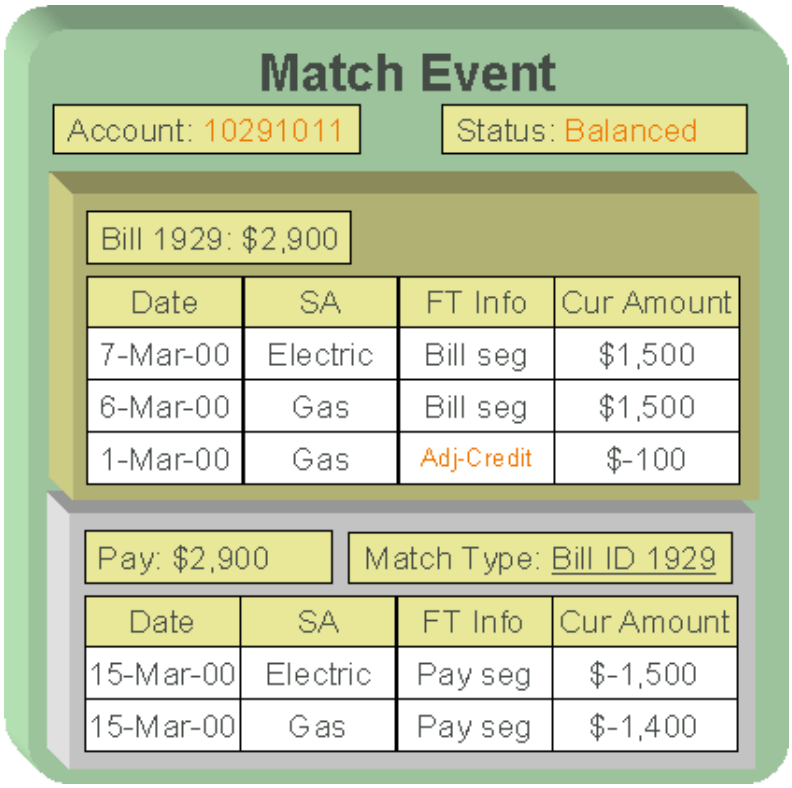
Note: The match type's distribution logic is not "hard coded". Because the match type's payment distribution logic is embedded in a plug-in algorithm, you can introduce new algorithms as per your company's requirements.

It's worth noting that payment *distribution* and *freezing* are two separate steps that typically happen in quick succession. The system's standard match event algorithms create the match event during payment distribution. This match event exists in the `open` state (because the payment segment's FT's have not yet been linked to the match event and therefore debit FT's do not equal credit FT's). The `open` match event references the debit FT's (the bill segments and adjustments) for which it pays. It is only at payment freeze time that the credit FT's (the payment segments) are linked to the match event thus allowing the match event to be become `balanced`.

If, at freeze time, the payment's credit FT's do not equal the debit FT's on the match event, the match event is left in the `open` state. An alert will appear on Control Central to highlight the existence of `open` match events (if the appropriate alert algorithm is plugged in the installation record). In addition, you can also set up a ToDo entry to highlight the existence of open match events.

Payments Are Matched To Debit and Credit FTs

While the above discussion dealt with the typical situation where the payment's credit FT's are matched against a bill's debit FT's, we want to point out that a payment's FT's may be matched against debit *and* credit FT's. Consider the following example:



Notice that:

- The $2,900 payment is distributed amongst two contracts.
- The FT's to which the payment segments are matched are both debit and credit FT's. Notice that the debit FT's (the bill segments) and the credit FT (the adjustment) sum to $2,900.

Credits may result in a situation where the total amount on a bill for an contract is negative. This would be the case if in the above example the credit adjustment were for $-1600 resulting in the total amount for the Gas contract on this bill to be $-100 (credit). Assume a full payment of $1400 is made towards this bill. The *Bill ID Match Type Algorithm* first allocates negative payment amounts to any contract credit amount on the bill being paid. It then carries over the

credit amount to pay off other bill amounts. In this example, a "negative" payment segment is created to match the $-100 credit of the Gas contract. Using the carried over credit a $1500 payment segment is created to match the $1500 debit of the Electric contract.

How Are Match Events Cancelled?

A user can cancel an `open` or `balanced` match event at any time. When a match event is `cancelled`, the event's FT's again affect arrears (and they can be associated with new match events). In other words, when a match event is `cancelled`, its FT's are released from the match event and become open-items.

In addition to manual cancellation, the system may automatically cancel a match event when the last of its payment FT's, if any, is cancelled (if you plug-in the appropriate FT freeze plug-in on your open-item customer classes).

For example, consider a match event that was created when a payment was made. If the payment is subsequently cancelled, the match event is also cancelled (thus releasing the match event's FT's) if no other payment FT's are linked to the match event. Please be aware that FT cancellation also causes a new match event to be created. This match event matches the original FT (the payment segment) and its cancellation FT. This means that the only "open items" that will exist after a payment is cancelled are the debit FT's that were originally paid.

Note: Reopening bills associated with automatic payment customers. While many payments are cancelled due to non-sufficient funds, please be aware that if you reopen a bill for which an automatic payment was created, the system will cancel the associated payment. If this payment is associated with a match event (because the account is an open-item account), the match event will be cancelled and a new match event will be created to match the original automatic payment with its cancellation details. This is necessary because a new payment will be created with the bill is subsequently completed and this payment's FT's will be matched to the bill's FT's.

Note: Canceling a payment can result in many match events being created. If a cancelled payment has multiple payment segments, a separate match event will be created for each payment segment.

While payment cancellation is the most common type of FT cancellation, be aware that bill segment or adjustment cancellation may also cause a new match event to be created. We don't necessarily want to always link the cancellation FT and its original FT to the same match event. For example, when the cancellation FT is swept on to the next bill it affects the next bill and not the original FT's bill. For cancellations that will *not* be swept on to the next bill (payment cancellation, cancellation of an adjustment that is not shown on bill, credit notes, and bill segment cancellation before the bill is completed) the system creates a new match event that matches the original FT and its cancellation FT. This way, neither FT affects aged debt. If the original FT was linked to an existing match event and no other FTs are left on this match event it is automatically canceled.

Current Amount Is Matched, Not Payoff

The system matches the current amount of financial transactions, not the payoff amount.

Fastpath: Please refer to *Current Amount versus Payoff Amount* for more information about current and payoff amounts.

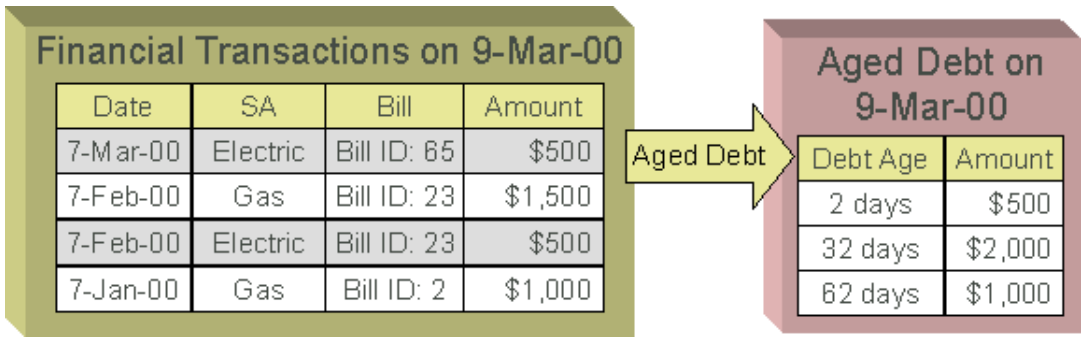
Disputing Items

Open-item customers may dispute FT's that they are not comfortable paying. For example, a customer who receives a bill with an anomalous charge may decide to dispute it.

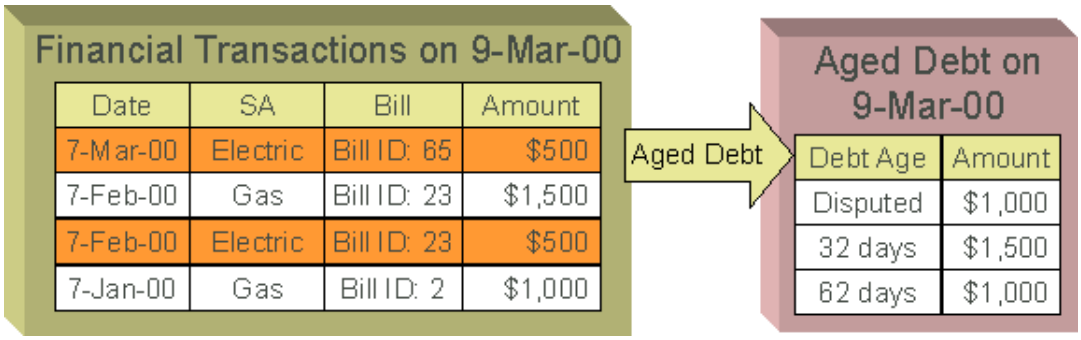
When an open-item customer disputes a charge, a user creates a match event and links the disputed FT(s) to it. This match event will be in the `open` state (because it does not contain FT's that sum to zero). In addition, the match event's "disputed switch" is turned on.

Note: Alerts. An alert is displayed on control central to highlight the existence of disputed match events (if the appropriate alert algorithm is plugged in). In addition, you can also set up a ToDo entry to highlight the existence of disputed match events.

While the dispute is being researched, the disputed amount will not affect aged debt, but it still forms part of the customer's balance. For example, consider the following unpaid financial transactions that exist for an account:



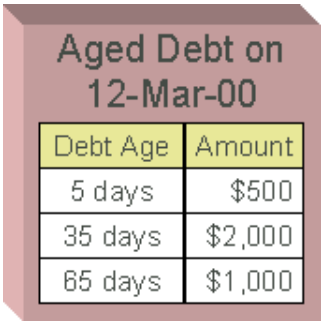
If the customer disputes the two electric bill segments, the customer's aged debt will look as follows:



Notice how a new category of debt appears - `Disputed`. Also notice how the 2 day old debt disappears and the 32 day old debt is reduced by the disputed amount.

The system shows disputed debt on Control Central. In addition, in all places where aged debt appears in the system, disputed debt is shown as a separate debt category.

If the dispute goes in your company's favor, the disputed match event should be `cancelled` (thus allowing the FT's to again impact aged debt). For example, if we assume 3 days have passed and the dispute match event is cancelled, the customer's aged debt will look as follows:



If the dispute goes in the customer's favor:

- You may decide to issue a credit note to cancel the offending bill or bill segment(s). As described above, the system in this case will automatically create new match events that match the original FTs with their cancellation FTs and cancel the disputed match event when the last item is unlinked from it.

- You may decide to cancel the offending bill segment(s) / adjustment(s). As described above, these cancellations are going to be swept on to the next bill. The system therefore will not automatically cancel the disputed match event. Notice that the cancellation effect of the disputed items is carried over on to the next bill. This means that the previously disputed items still need to be paid.

Note: Cancel / rebill. If you cancel / rebill an offending bill segment, both the cancel and the rebill will become open-items that will be matched when the next bill is paid.

- You may decide to issue an adjustment to counter the effect of the disputed FT's. In this situation, you would simply link the adjustment FT to the disputed FT's (thus allowing the match event to become `balanced`). It is important to use in this case an adjustment that does not show on bill.

Promise To Pay

You create a promise to pay when a customer agrees to make one or more scheduled payments to satisfy past (or future) debt. These payments cannot be matched to open items because it is unlikely that debit FT's exist that equal the amount of each scheduled payment. However, you must specify a match type on all payments made by open-item customers. Therefore, a conundrum exists - the system requires a match type on payments made by open item accounts, but payments made for a promise to pay cannot be matched to existing FT's. This conundrum is solved by the fact that match types do not have to specify an override payment distribution algorithm. The customer class's standard distribution algorithm is used for payments that reference such a match type.

You may wonder how these payments will eventually get matched to open items? If ALL payments associated with a promise to pay occur before the next bill is paid (or if the promise to pay exists to satisfy future debt), these payments will be swept onto the match event that is created when the customer pays their next bill. However, if the promise to pay exists to payoff historical debt and this debt has not been entirely paid by the time of the next bill, an unmatched event will exist when the customer pays their subsequent bills (if the payment amount doesn't match the amount of new charges on the bill). Why? Because, the customer is not paying the entire amount of the bill and therefore the system will not be able to match the payment to open items. If this occurs, we recommend canceling the match events that are created when the customer pays their subsequent bills. When the customer finally pays off all outstanding debt, the system will create a single match event that will contain all payments and bill segments.

Overpayments

An overpayment, by definition, does not "match" to open items. However, the match type algorithms supplied with the base package will result in a `balanced` match event if an overpayment is made. The following points explain how this is achieved:

- The base package's match type algorithms will distribute the payment until the customer's current debt is satisfied.
- The amount of the overpayment will be kept on a separate contract (this only happens if you plug-in the appropriate Overpayment Distribution algorithm on your customer classes). Refer to *Overpayment Segmentation* for more information.
- When the payment is frozen, the payment segments that satisfy current debt will be matched against their respective open-items. The payment segment used to book the overpayment (on the overpayment contract) will not be matched.
- When future bills are completed, the credit balance on this "overpayment contract" will be transferred to the "real contract's" when future bills are completed (if you have plugged in the appropriate bill completion algorithm on the overpayment contract's contract type). If the overpayment satisfies all newly calculated charges, a match event is created that matches the new charges against the funds transferred from the overpayment contract. Refer to *When Are Match Events Created* for information about how the system creates match events at bill completion time when the new charges on the bill are satisfied by other credits (overpayments, deposit refunds, etc.).
- At some point in the future, the overpayment will be exhausted (i.e., all funds will be transferred to "real contract's"). At that point in time, the overpayment contract will close (assuming you set up the overpayment contract's contract type as a "one time"). At close time, the system creates a match event that matches the

original overpayment payment segment with the adjustments that were used to transfer funds to the "real contract's". Refer to *When Are Match Events Created* for information about how the system creates match events when a contract closes.

Setting Up The System To Enable Open Item Accounting

The following section provides an overview of how to enable open-item accounting.

Match Type Setup

The number of match types that you will need is dependent on the number of ways you want payments to be matched to open items. At a minimum, you will probably need the following match types:

- **Bill ID.** This match type should reference an override payment distribution algorithm that distributes the payment based on the bill ID specified on the payment (in match value). Refer to *Payments And Match Events* for more information.
- **Contract ID.** This match type should reference an override payment distribution algorithm that distributes the payment based on the Contract ID specified on the payment (in match value). Refer to *Payments And Match Events* for more information.
- **Promise to Pay.** This match type should NOT reference an override payment distribution algorithm (if this algorithm is blank, the customer class's payment distribution algorithm is used). Refer to *Promise To Pay* on page 231 for more information.

Match Event Cancellation Reason Setup

The number of match event cancellation reasons that you will need is dependent on the number of ways your organization can justify the cancellation of a match event. At a minimum, you will probably need the following match event cancellation reasons:

- **FT Cancellation.** This cancel reason should be referenced on the Customer Class FT Freeze algorithm that is responsible for canceling match events when one of its financial transactions is cancelled.
- **Incorrect Allocation.** This cancel reason should be specified by users when they cancel match events that were created by the system erroneously.

Customer Class Setup

The following points describe *customer class* oriented set up functions:

- Turn on the open-item accounting switch.
- Set up the following algorithms for each division:
  - Specify a **payment freeze** algorithm that causes a payment's FT's to be linked to the match event that was created when the payment was distributed. Refer to *Payments And Match Events* for more information.
  - Specify a **FT freeze** algorithm that causes match events to be cancelled (and a new match event to be created) when a FT is cancelled. Refer to *How Are Match Events Cancelled* for more information about cancellation.
  - We strongly recommend specifying an **overpayment** algorithm that causes overpayments to be segregated onto an "excess credit / overpayment" contract. Refer to *Overpayments* for more information.

Overpayment Contract Type Setup

Specify a **bill completion** algorithm that causes the credit amount on overpayment contract's to be transferred to newly create debt (created when the bill is created). This algorithm transfers an overpayment contract's balance to regular contract's and creates a match event if the overpayment covers the entire bill. Refer to *Overpayments* for more information.

Installation Record Setup

Specify an **automatic payment** algorithm that causes a match event to be created when automatic payments are created for open-item accounts. The base package algorithm will do this for you if you specify the appropriate parameter on the algorithm. Refer to *APAY-CREATE* for more information about this algorithm.

If you want a Control Central alert to highlight when the current account has any open match events, plug in the appropriate **control central alert** algorithm on your installation record. Refer to *C1-OPN-MEVT* for more information about this algorithm.

If you want to enable manual pay segment distribution for open item accounts, along with other functions, you will need to plug in an installation algorithm for bill balance calculation. Refer to *C1-OI-BI-AMT* for more information about this algorithm.

To Do Entry Setup

Two ToDo types are supplied with the base package:

- **TD-MODTL**. This ToDo type highlights the presence of open, disputed match events.
- **TD-MONTL**. This ToDo type highlights the presence of open, non-disputed match events.

Each of the above ToDo types should be configured with the roles that work on entries of each type.

In addition, the account management group and/or divisions from which the default roles are extracted should be updated to define the role that should be defaulted for each of the above ToDo types.

Fastpath: Refer to *The Big Picture Of To Do Lists* for more information about ToDo lists.

Setting Up Match Types

Most payments are distributed amongst contracts using the payment distribution algorithm specified on the payment's account's customer class. This algorithm decides how to distribute a payment amongst an account's existing debt *if the customer doesn't specify how the payment should be distributed*.

A customer can specify how a payment is distributed by specifying a match type and match value on their payments. Consider the following examples:

- Customers that are subject to open-item accounting (this is defined on the account's customer class) tell the system exactly which debt is covered by their payments. For example, an open-item customer might make a payment in respect of bill ID `123919101919`.
- Even non open-item customers can direct payments to specific contract's. For example, the system allows a balance-forward customer's payment to be directed to a specific contract (however, they cannot direct payments to specific bills as only open-item customers can do this).

Match types are used to define the specific type of debt that is covered by a payment. The match type contains the algorithm that effectively overrides the standard payment distribution algorithm defined on the account's customer class.

Note: Background information. Please refer to *Payments And Match Events* and *Match Type Setup* for more information about how match types are used.

To set up match types, select **Admin Menu, Match Type**.

Description of Page

Enter an easily recognizable **Match Type** and **Description**.

Define the **Pay Dist Override Algorithm** used to distribute payments that reference this match type. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).

- On this algorithm, reference an Algorithm Type that overrides the normal payment distribution algorithm.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_MATCH_TYPE*.

Setting Up Match Event Cancellation Reasons

When a match event is canceled, a cancel reason must be supplied.

Note: Background information. Refer to *How Are Match Events Cancelled?* and *Setting Up Match Event Cancellation* for more information about cancellation.

To set up match event cancellation reasons, select **Admin Menu, Match Event Cancel Reason**.

Description of Page

Enter an easily recognizable **Match Event Cancel Reason** and **Description**.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_MEVT_CAN_RSN*.

Fund Accounting

The topics in this section provide background information about fund accounting.

Note: This section is only relevant for some organizations. The system configuration requirements described in this section are only relevant if your organization practices fund accounting. If your organization does not practice fund accounting, you need only indicate such on the *Installation Record*; no other setup is required.

Fund Accounting Overview

Non profit organizations in general, often use a form of accounting different from that used by for-profit corporations. Non profit organizations typically practice fund accounting, whereas corporations practice corporate accounting.

Regulations or other restrictions may require a municipal utility to account for the finances of each of its departments as a separate entity.

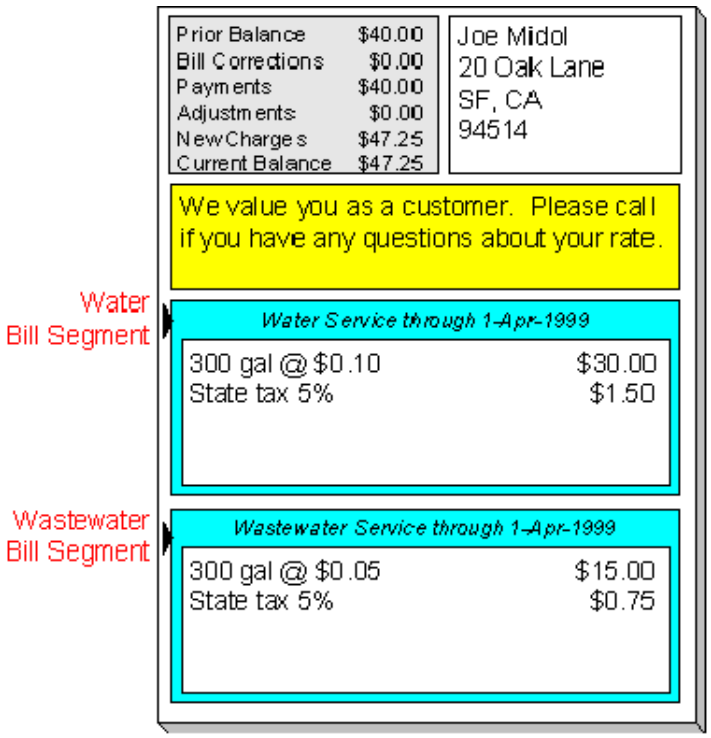
A fund is an accounting entity with its own self-balancing set of accounts. Each fund has its own "sub general ledger" with its own chart of accounts, and within each fund, its debits equal its credits at all times. This allows the utility to report on the financial state of each fund independently.

In addition to having a fund for each department, there is also a general fund, which is used to handle inter-fund transfers as well as shared accounts.

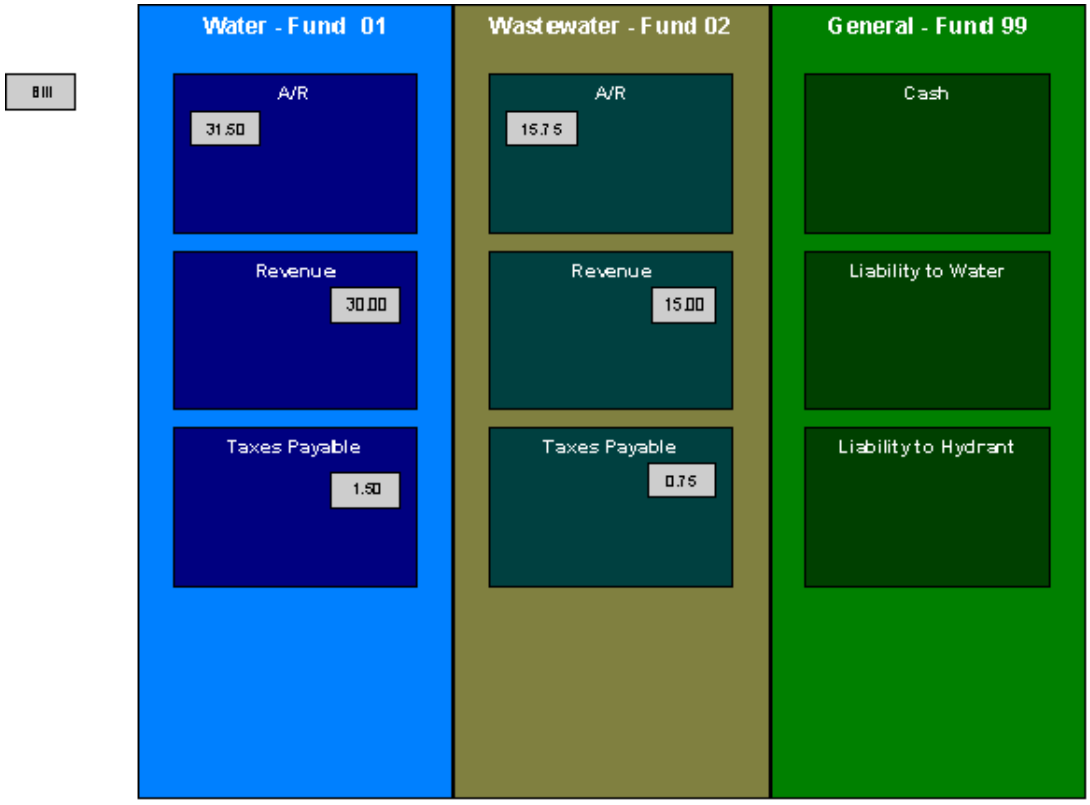
Fund Accounting Example

In addition, with fund accounting, there is always a general fund (fund 99).

Assume the following bill is generated.



The bill would produce the following GL entries:

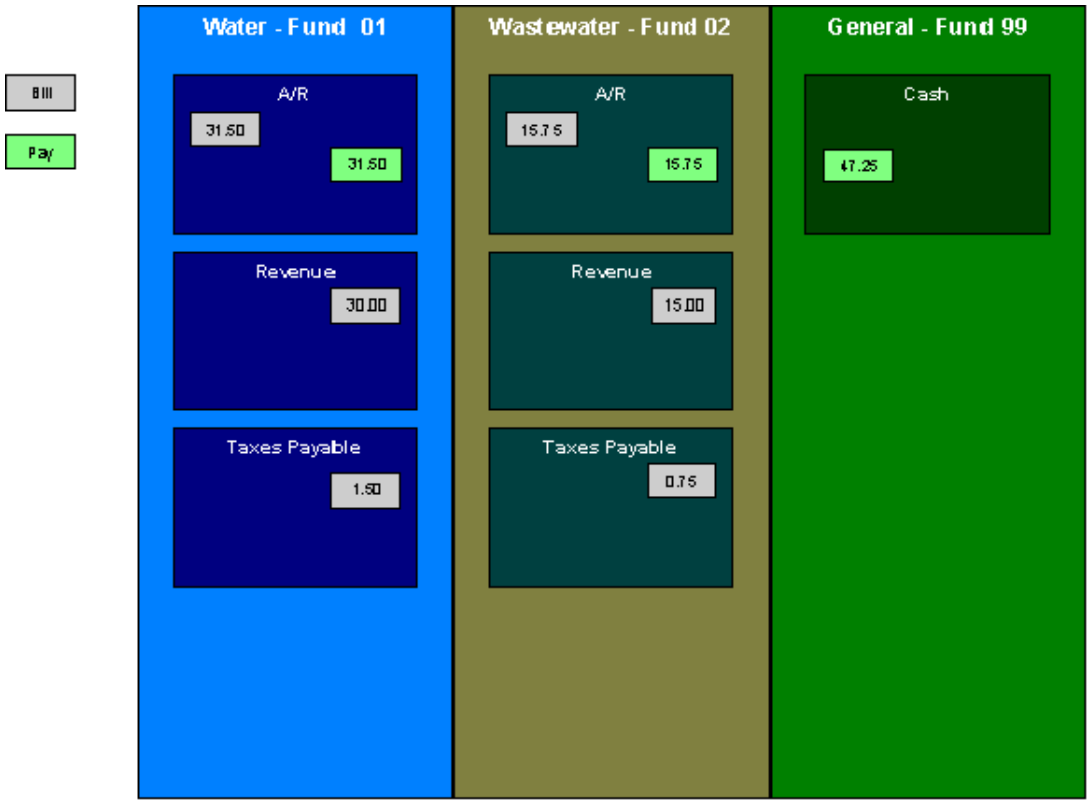


For each fund, the GL details of the bill will include a debit to the accounts receivable (A/R) account and credits to the revenue and taxes payable accounts. In organizational terms, each department is owed a portion of the overall bill

by the customer, part of which is sales by the department and part of which is owed to the taxing authorities by the department. Each fund is balanced.

Note that the accounting could be identical under corporate accounting if each service is its own division with its own chart of accounts.

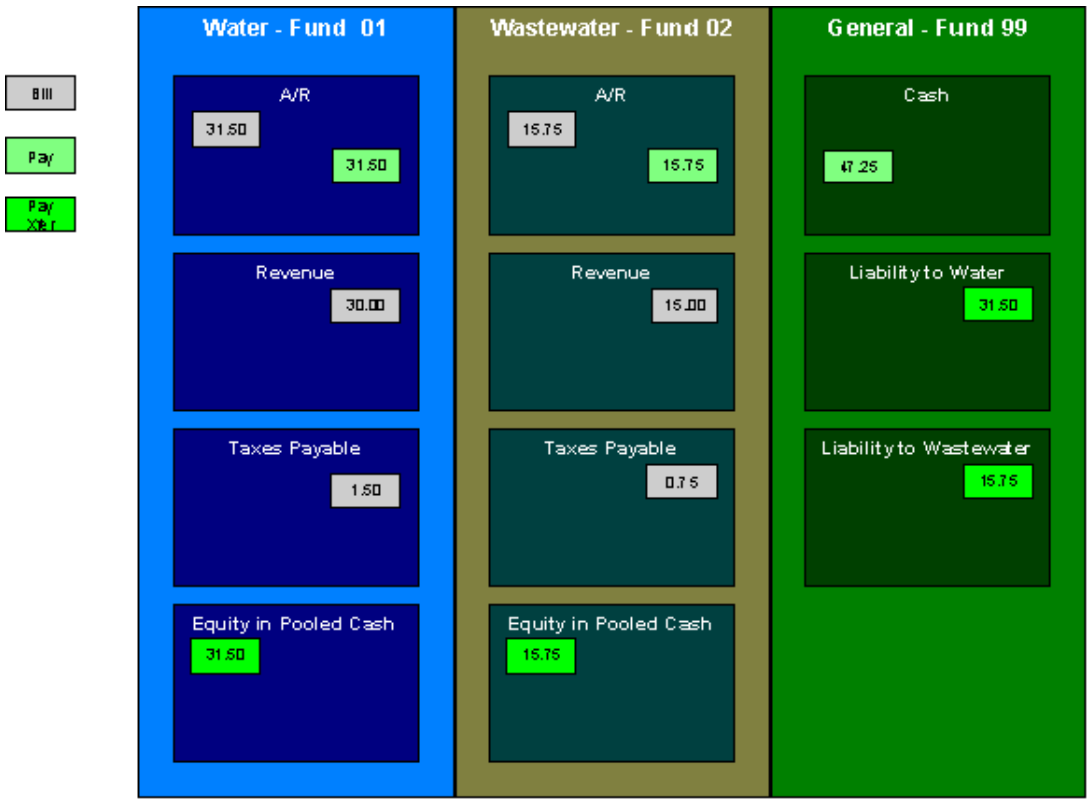
The following diagram illustrates the initial GL accounting that would occur when the payment arrives:



The general cash account is debited, and the departmental funds' A/R accounts are credited. In other words, the cash is held as a whole but the receivables are reduced for the individual departments.

If the accounting were left in this state, the fund accounting principal - that each fund represents an independent entity with a self-balancing chart of accounts - would be violated. This violation is caused due to the fact that cash is recorded on the general fund, not the departmental funds, causing the general fund to have an excess debit and the departmental funds to have an excess credit.

From an organizational viewpoint, to make each department whole, the departments need to note what portion of the cash they own. The following diagram illustrates this point.



To maintain a balance of debits and credits within each fund, the departmental funds have an "equity in pooled cash" (EPC) account and the general fund has a liability account for each departmental fund. In addition to debiting the general fund's cash account and crediting the departmental funds' A/R accounts, the departmental funds' EPC accounts are debited and the general funds liability accounts are credited.

And so, with the additional GL entries, all funds have matching debits and credits.

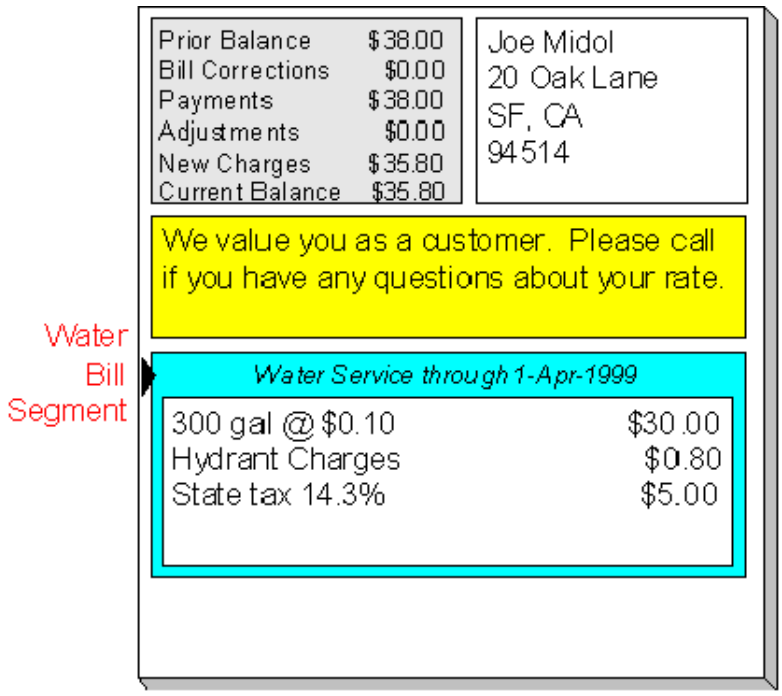
An Example Of A Bill Segment That References Multiple Funds

Consider a municipal utility that primarily supplies water service but is also responsible for maintaining the city's fire hydrants. The costs for fire hydrant maintenance are borne by the water customers and make up just a small portion of the overall bill. These costs are simply added to the water bill as a line item. The utility has two departments:

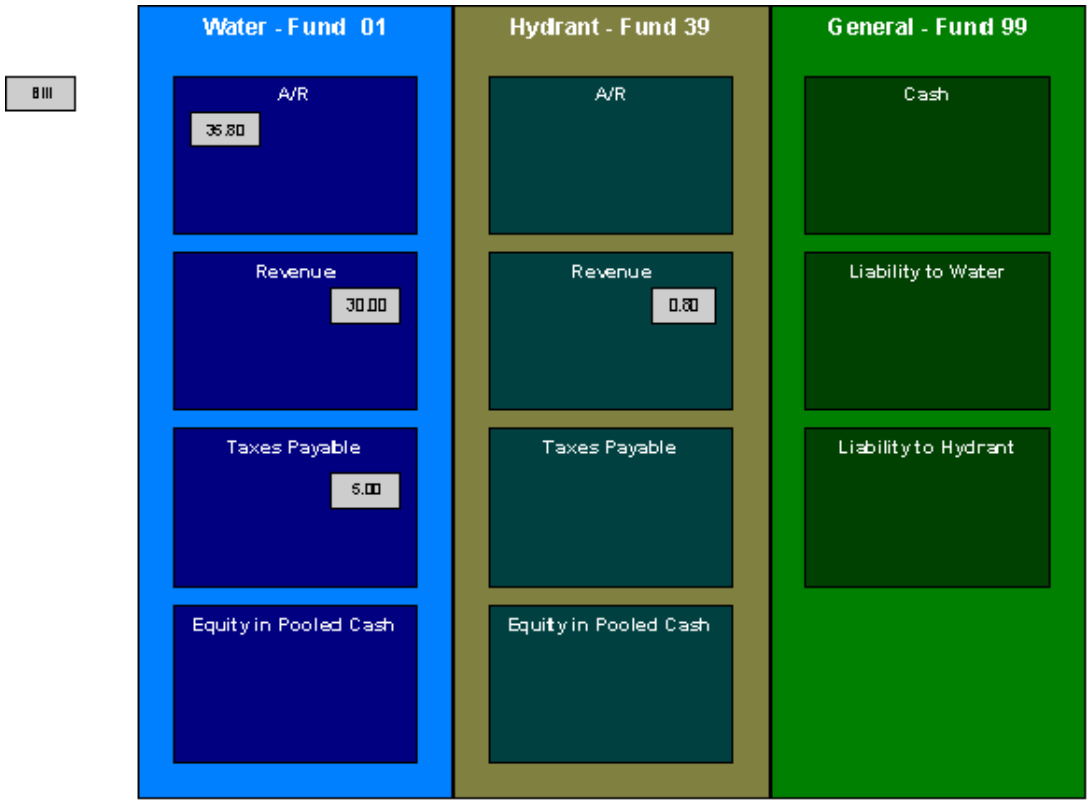
- Water service (fund 01)
- Hydrant maintenance (fund 39).

In addition, there is a general fund (fund 99).

Assume the following bill is generated for water and hydrant services.



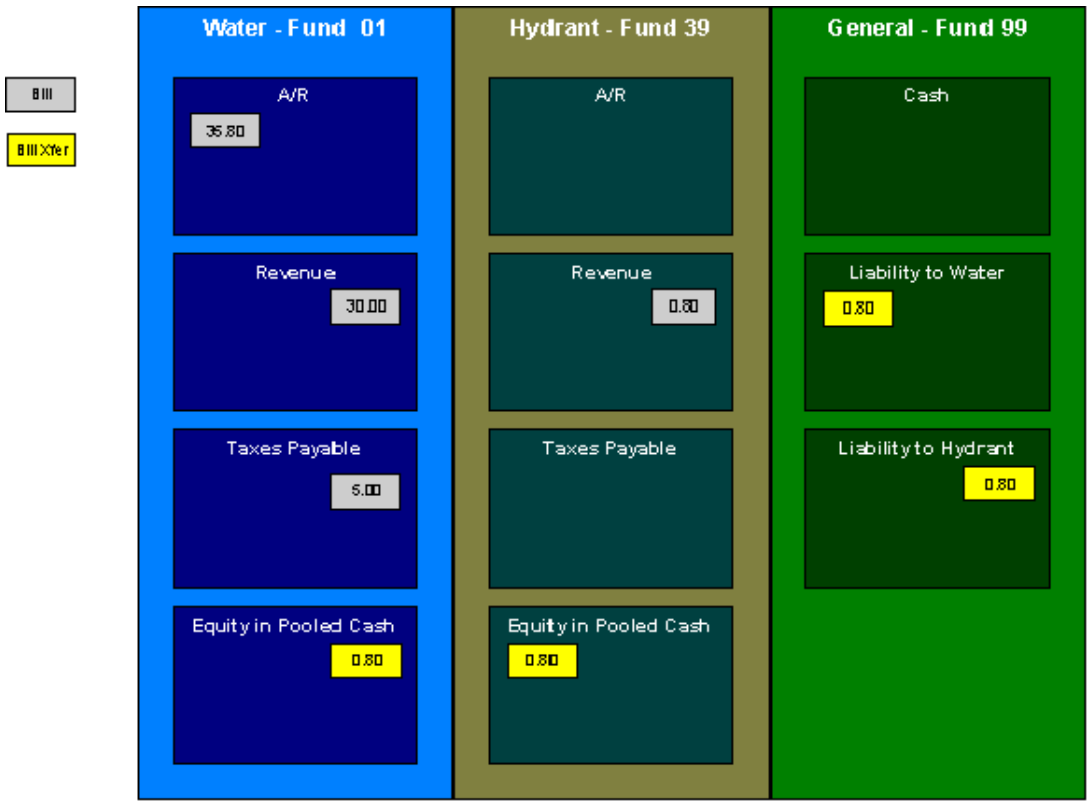
The following diagram illustrates the initial GL entries for the bill:



In accounting for the bill, the water fund's A/R is debited, the water and hydrant funds' revenue accounts are credited, and the water's taxes payable account is credited.

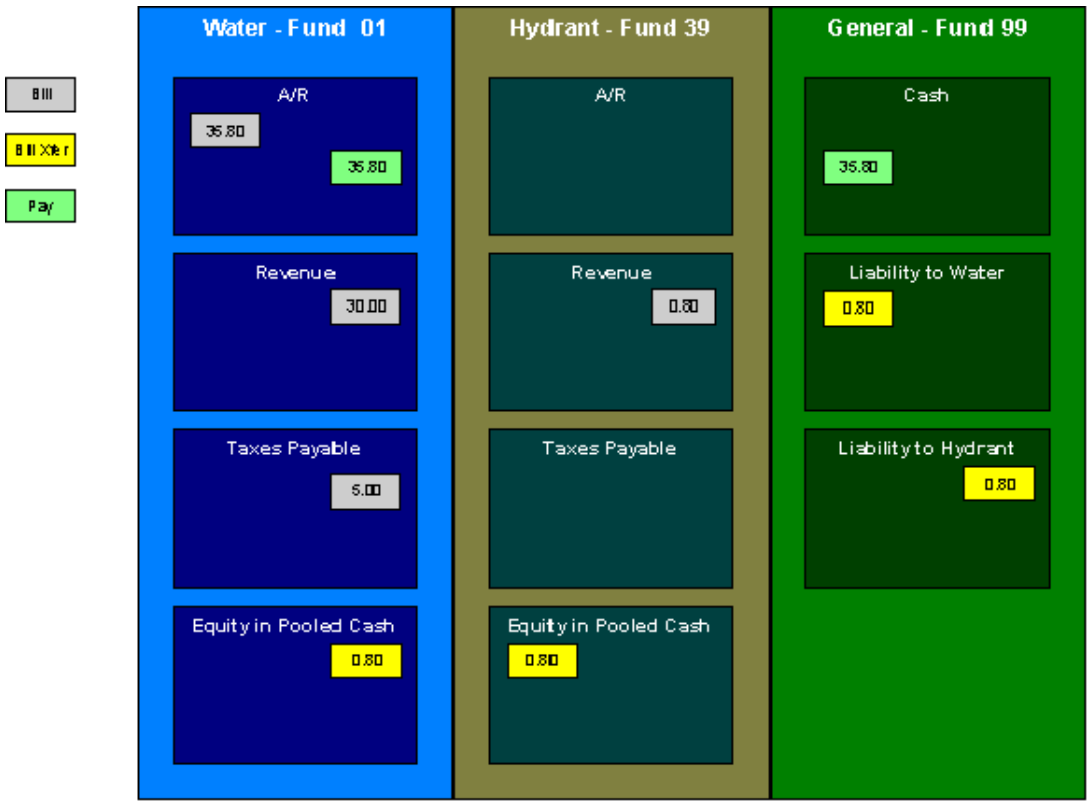
If left at this, the funds would be out of balance; the water fund would have an overall excess debit and the hydrant fund would have an equal excess credit. In organizational terms, the hydrant fund has recorded sales but that amount is recorded as being owed to the water department.

To balance each department, the water department accepts the responsibility for collecting the hydrant charges from the customer but immediately remunerates the charges to the hydrant fund.

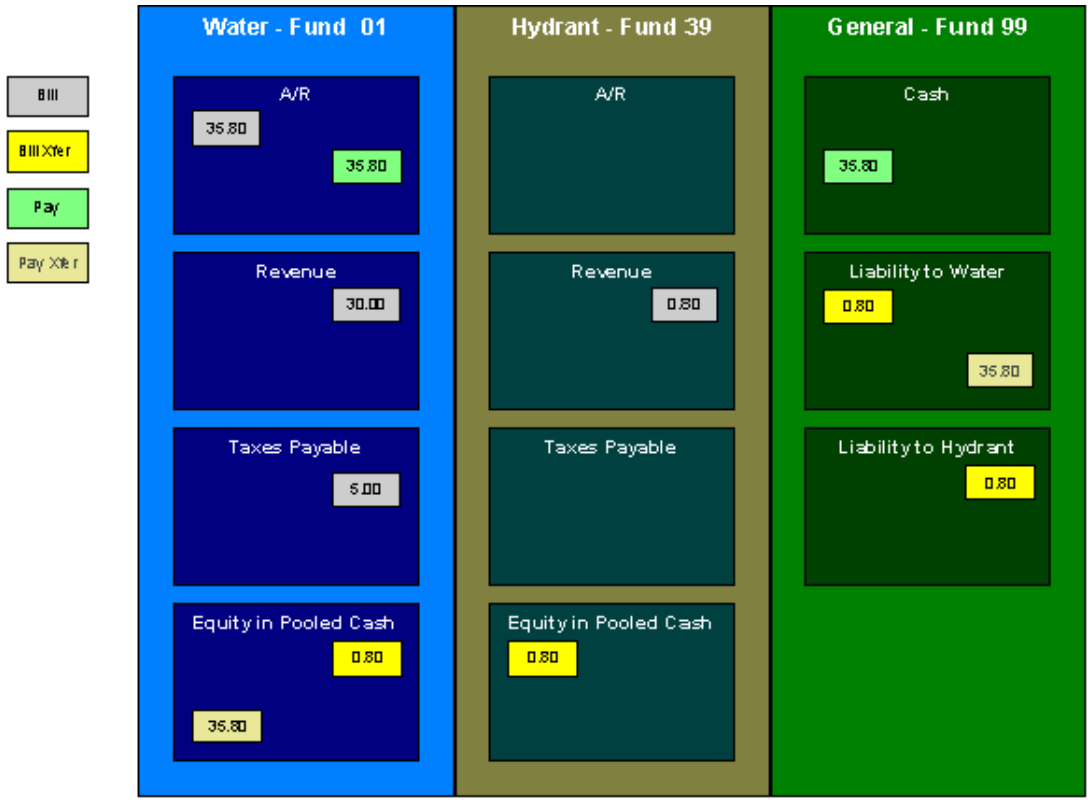


This transfer is done using the general fund. The water fund's EPC account is credited and the liability to water is debited with the amount of the hydrant revenue. Also, the hydrant fund's EPC account is debited and the general fund's liability to hydrant account is credited by the hydrant revenue. In effect, the water department owes the hydrant charges to the general fund, and the general fund owes the hydrant charges to the hydrant fund.

The following diagram illustrates the initial GL accounting that would occur when the payment arrives:



When the payment arrives, the cash is debited to the general fund's cash account, and the water fund's A/R is relieved. Again, the funds would be unbalanced if left in this condition; the water fund would have an excess of credits and the general fund would have an excess of debits.



To maintain each fund's balance of debits and credits, the general fund's liability to the water fund is credited by the amount of the department's share of the cash, and the water fund's EPC is debited. Note that the payment has no effect on hydrant fund's EPC and the general fund's liability to the hydrant fund. The hydrant department "received" its money from the water department when the bill was created.

And so, all funds have matching debits and credits.

Accounting Method Is Defined On The Installation Options

You must turn on a switch on the *Installation Record* to enable fund accounting.

Fund Controls Fund-Balancing Entries

There are two levels of debit and credit balancing in fund accounting. There is the balancing required by double entry accounting: the total debits in the entire GL must equal the total credits. This is required regardless of whether fund or corporate accounting is used. The distribution codes for these entries come from varying sources, depending on the type of financial event.

Fastpath: Refer to *The Source Of GL Accounts On Financial Transactions* for information on the sources of the distribution codes.

The second level of balancing is specific to fund accounting. Within each fund-not just across the GL-the total debits must equal the total credits. The original distribution code from the financial event has a fund specified. For example, a bill would cause a debit to a fund's A/R distribution code, and included in that A/R distribution code is the fund. It is the definition of the fund that specifies whether fund-balancing entries are required and provides the distribution codes for these entries.

For a departmental fund, the fund-balancing debit and credit would be specified. When a debit is applied to a departmental fund's GL account, an additional account (typically the general fund's liability to the departmental fund) is debited and an account (typically the departmental fund's EPC) is credited. When a credit is applied to a departmental fund's account, an additional account (typically the general fund's liability to the departmental fund) is credited and an account (typically the department's EPC) is debited.

For the general fund, no fund-balancing debits and credits are specified.

Building Fund - Balancing GL Details

Building the GL details for a financial event is a two-step process.

- First, the system generates the regular GL details for a financial transaction (FT). This is done regardless of whether corporate or fund accounting is used.
- Second, if fund accounting is activated (by turning on a switch on the *Installation Record*), the system analyzes the distribution code on each GL detail associated with the FT. If a *fund* is specified on a distribution code, the system checks the definition of the fund. If fund-balancing entries are specified on the fund, two additional GL entries are added to the FT:
- An offsetting entry to the Equity in Pooled Cash account is created for the departmental fund (e.g., if the FT is debiting a given fund, an offsetting credit is created in the funds EPC account).
- Another entry to the departments Liability account is created for the general fund.

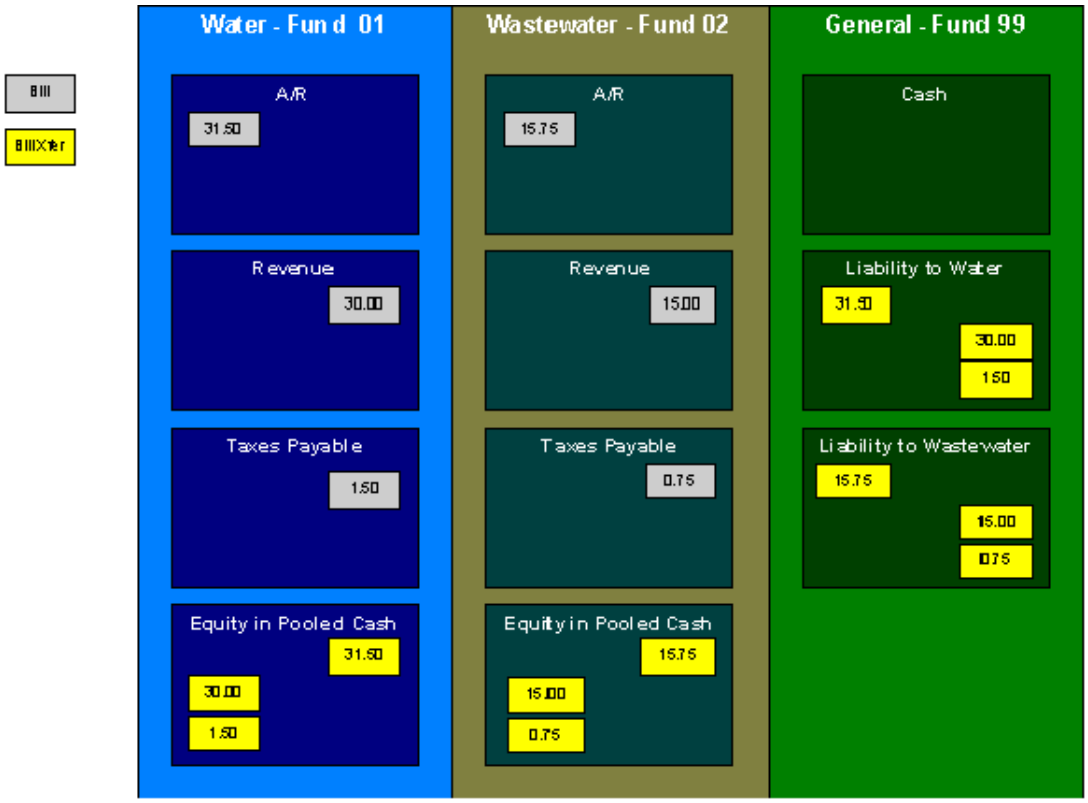
The result is a consolidated set of GL entries for the FT, incorporating the regular entries as well as the fund-balancing entries.

The topics in this section illustrate the generation of the GL details for the earlier examples.

FTs Whose GL Details All Reference The Same Fund Do Not Impact the General Fund or EPC Accounts

In *Fund Accounting Example*, where the bill's bill segments reference a single fund, the system creates a fund-balancing GL entry for each GL entry applied to a departmental fund:

- A debit to a departmental GL account triggers a debit to the general fund's liability-to-departmental-fund account and a credit to the departmental fund's equity-in-pending-cash account.
- A credit to a departmental GL account triggers a credit to the general fund's liability-to-departmental-fund account and a debit to the departmental fund's equity-in-cash account.

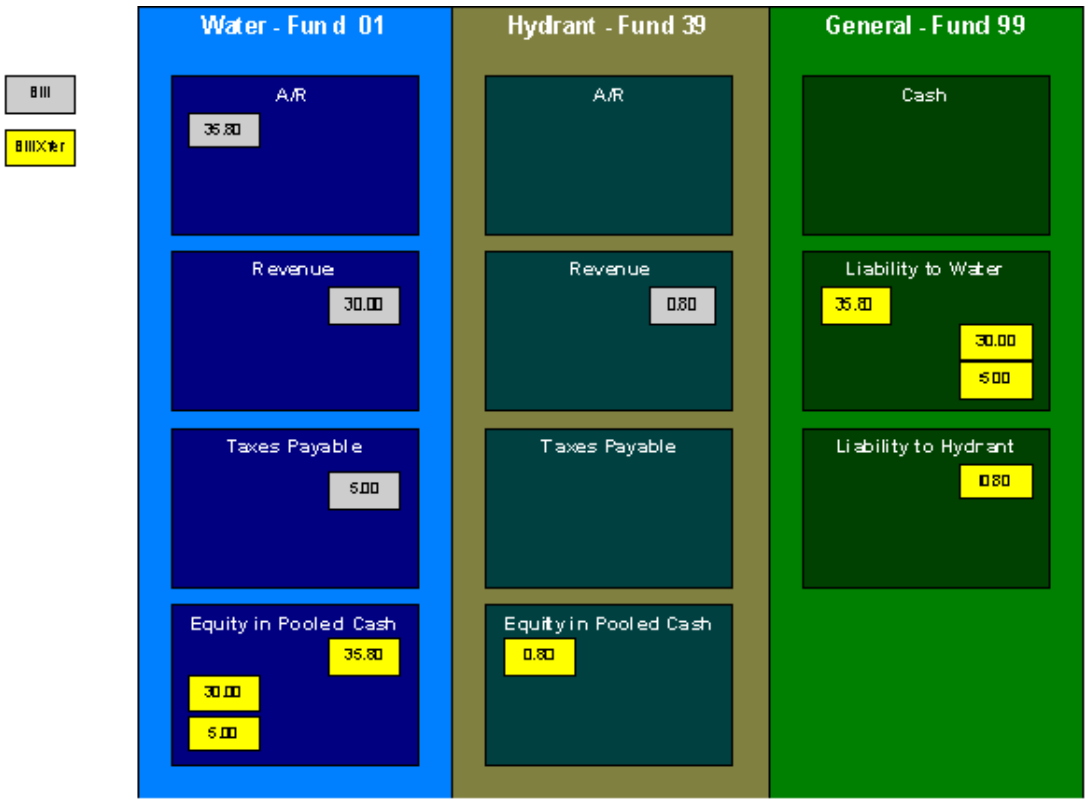


The net effect on the individual equity-in-cash and general fund's liability accounts is zero because the debits and credits net to zero for each GL account. In other words, the yellow boxes net to zero and therefore fund accounting does not impact the bill segment's financial transactions. Refer to *Fund Accounting Example* for the resulting consolidated GL entries.

An FT Whose GL Details Reference Multiple Funds

In *An Example Of A Bill Segment That References Multiple Funds*, where the bill's bill segments reference multiple funds (water and hydrant), the system also creates fund-balancing GL entries for the financial transaction:

- A debit to a departmental GL account triggers a debit to the general fund's liability-to-departmental-fund account and a credit to the departmental fund's equity-in-pending-cash account.
- A credit to a departmental GL account triggers a credit to the general fund's liability-to-departmental-fund account and a debit to the departmental fund's equity-in-cash account.



The net effect of the bill on the GL is that the water fund's EPC has a credit of $0.80, the hydrant fund's EPC has a debit of $0.80, the general fund's liability to the water fund has a debit of $0.80, and the general fund's liability to the hydrant fund has a credit of $0.80. Note that, overall, the general fund's overall liability to the departmental funds nets to zero. Refer to *An Example Of A Bill Segment That References Multiple Funds* for the resulting consolidated GL entries.

Setting Up The System To Enable Fund Accounting

The following section provides an overview of how to enable fund accounting.

Turn On Fund Accounting

On the *Installation Record*, indicate that fund accounting is `Practiced`.

Defining Funds

A fund must be setup for each specific fund in your organization. Don't forget to also set up a fund for the general fund. Navigate using **Admin Menu, Fund**.

Description of Page

Enter a **Fund** and a **Description** to identify the fund.

If this fund is used to balance other funds or to hold cash, indicate a **Fund Type** of `General`, otherwise indicate that it is `Specific`.

If the fund type is `Specific`, specify the **Equity Distribution Code** and **Liability Distribution Code**. These codes are used to balance financial transactions that span funds. The equity distribution code should belong to the same **Fund** as the one you are defining. The liabilitydistribution code should belong to the general fund.

Distribution Codes Must Include Fund ID

All of your distribution codes must include their respective fund ID.

Fastpath: **For more information**, refer to *Setting Up Distribution Codes*.

Update Your Funds With Their Respective Equity and Liability Distribution Codes

After distribution codes have been setup, you must update your funds to indicate the equity and liability accounts used to balance inter-fund financial transactions.

Other Financial Transaction Topics

Various topics about financial transactions are discussed in this section.

The Source Of GL Accounts On Financial Transactions

The following table lists the major financial events, their standard accounting, and the source of distribution codes used to derive the GL accounts sent to your general ledger.

Financial event	GL Accounting	Source Of Distribution Code
Create a normal financial services bill segment. Bill Segment FT Algorithm is Payoff Amt = Bill Amt / Current Amt = Amt Due	Debit: A/R	Contract Type
	Credit: Revenue / Taxes Payable	Rate Component
Create a bill for company usage. Bill Segment FT Algorithm is Payoff Amt = 0 / Current Amt = 0	Debit: Company Usage Expense	Contract Type
	Credit: Revenue / Taxes Payable	Rate Component
Create a bill for charity. Bill Segment FT Algorithm is Payoff Amt=0 / Current Amt = Bill Amt	N/A - charity bills have no effect in the GL	N/A
	N/A	N/A
Create a payment segment for a normal financial services contract	Debit: Cash	Bank Account on the Tender Source of the Tender Control for the Payment Segment's Tender.
	Credit: A/R	Contract Type
Create a payment segment for a charitable contribution contract	Debit: Cash	Bank Account on the Tender Source of the Tender Control for the Payment Segment's Tender.
	Credit: Charity Payable	Contract Type

Financial event	**GL Accounting**	**Source Of Distribution Code**
Create a payment segment for auto-pay at bill completion time	Debit: Cash	Bank Account on the Tender Source on the Auto-pay Route Type of the Auto-pay Source.
	Credit: A/R	Contract Type
Canceling a payment	Debit: A/R	Contract Type
	Credit: Cash	Bank Account specified by the user on the cancel tender page. Note that this defaults to the original tender's bank account.
Create an adjustment to levy a charge	Debit: A/R	Contract Type
	Credit: Revenue	Adjustment Type

The bottom line is as follows:

- If a bill segment has a financial effect, the distribution code to debit comes from the distribution code on the contract type, the distribution code to credit comes from the rate component(s) used to calculate the bill segment.
- Payment segments always have a financial effect; the distribution code to debit comes from the bank account on the tender source of the tender control of the tender, the distribution code to credit comes from the contract type.
- If an adjustment has a financial effect, the distribution code to debit and credit comes from the contract type and adjustment type. If the adjustment is positive (i.e., the customer owes your organization more money), the distribution code to debit comes from the contract type; the distribution code to credit comes from the adjustment type. Vice versa if the adjustment is negative.

Chapter 4

Defining Customer Options

Topics:

The definition of a customer is someone (or something) with financial obligations with your company. These obligations ensue because the customer has agreed to purchase goods or services at an agreed price.

You may be surprised to learn that there is no "customer" record in the system. Rather, the system subdivides person information into the following records:

- **Person.** The person record holds demographic information about your customers and every other individual or business with which your company has contact. For example, in addition to customers, person records also exist for contractors, accountants at corporate customers, guarantors of customers, collection agencies, etc.
- **Account.** Accounts are the entities for which bills are produced and therefore you must create at least one account for every person who has financial obligations with your company. The account record contains information that controls when the bills are created and how the bills are formatted.
- **Contract.** Think of a contract as a contract between your company and the customer. The contract contains the terms and conditions controlling how the bill details are created. Every account will have at least one contract (otherwise, nothing will appear on the account's bills).

Before you can define persons, accounts, and contracts, you must set up the control tables defined in this section.

Fastpath: For more information about how persons and accounts are used by your customer service reps, refer to *Understanding The "V"*.

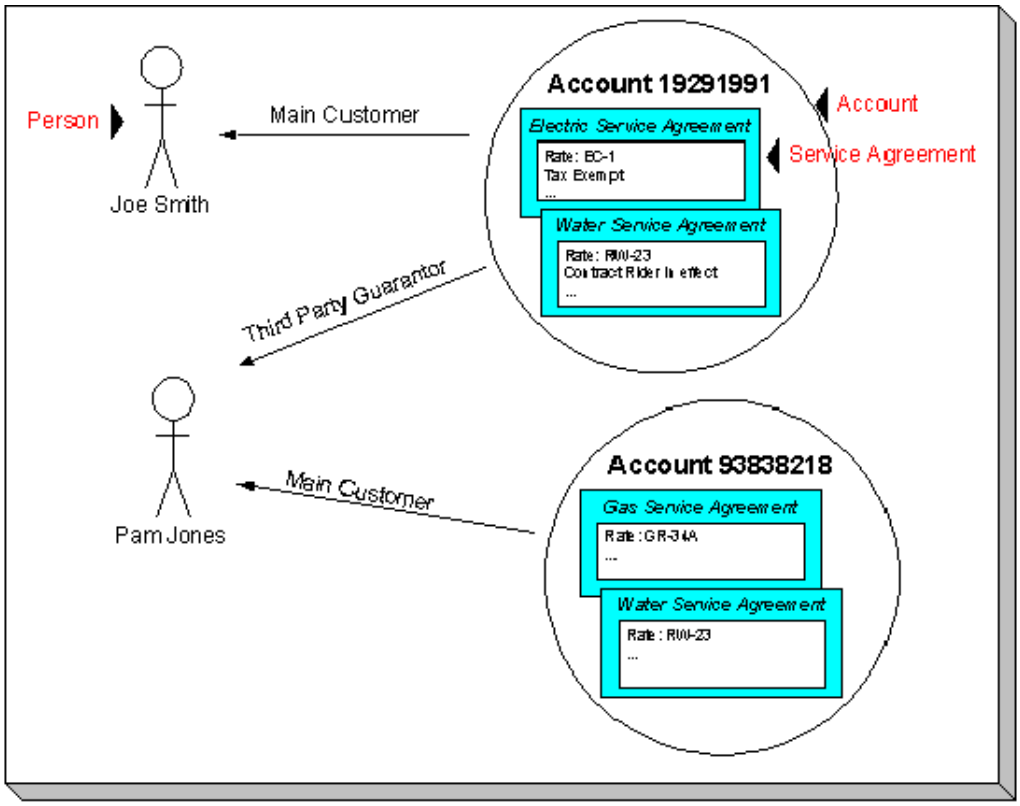
Note: The tables in this section are only some of many tables that must be set up before you can bill your customers for the service(s) they consume. In this section, we limit the discussion to those tables that control basic demographic and financial information. In later sections, we describe the tables that control other billing-related functions like bill creation algorithms and rates. It is only after all of these tables are set up that you will be able to generate bills and record payments.

Customer Overview

This section describes how the person, account, and contract records are used to record your customers' demographic and billing options.

A Simple Example Of Two Customers

The following picture illustrates two customers: Joe Smith and Pam Jones. Joe is the "main customer" on his account. Pam is the "main customer" on her account. Pam is also the "third party guarantor" on Joe's account.



Persons

Person records hold demographic information about the individuals and businesses with whom your organization communicates. Demographic information includes phone number(s), names and aliases, identification numbers, etc.

In the above example, 2 person records would be needed; one for Pam Jones and another for Joe Smith.

A new person is added when you first have contact with a person; the person does not have to be a customer before it is added. So, for example, if your company is starting a new marketing campaign, you can add information about potential customers the moment they are identified.

Note: Businesses are persons too. In addition to humans, you use person records to maintain basic information about the businesses with which your organization has contact.

Fastpath: For a description of the control tables that must be set up before you can define a person, refer to *Setting Up Person Options* .

Accounts

An account is analogous to an account at a bank:

- A person or business with no financial dealings with a bank will have no account (but the bank may choose to keep demographic information about the person as part of their marketing efforts). The exact analogy exists in this system.
- Individuals with financial dealings with a bank will have one or more accounts. The number of accounts is up to the customer. The exact analogy exists in this system.

A simple way to determine the number of accounts a customer will have is to ask "how many bills do they want each period?" because a customer receives one bill for each account. For example, a conglomerate that owns several offices may want their charges to appear on a single bill rather than have a separate bill for each office. This customer would have a single account.

Account ID Is Non-Intelligent

The unique number of an account is referred to as the "account ID". You are probably very comfortable with this concept. You may, however, have difficulty dealing with the fact that the account id in this system has no intelligence built into it (e.g., many systems include the bill cycle in the account id). In this system, the account ID is a random, system-assigned value.

Because the account ID contains no meaning, it can remain with a customer for life, regardless of where they live, when they are billed, the type of service they receive, etc. This is important because it means that all of the financial history linked to the account remains with the customer for life.

Note: Technical note. The non-intelligence of the account ID is also important from the perspective of the parallel processing that takes place when the system creates bills. Because the collection of accounts to be billed in any given bill cycle will be randomly distributed through the number spectrum, the system can distribute account number ranges to parallel threads and each thread will process roughly the same number of accounts.

Account / Person Cross Reference

A person may be linked to zero or more accounts. A person won't be linked to an account when they have no financial relationship with your organization. A person will be linked to multiple accounts when they have financial relationships with more than one account.

An account must reference at least one person (i.e., the main customer), but may reference an unlimited number of individuals. Multiple persons are linked to an account when several parties have some type of financial relationship with the account (e.g., third party guarantors, account contact, bill copy recipients, etc.).

When Is An Account Created?

A person can exist without an account until such time as the person formally requests the commencement of service. The moment the customer requests service, an account must be created (and the person must be linked to the account).

When Is An Account Expired?

Accounts never expire. Once a customer has an account, the account remains in the system forever. Linked to the account are contracts that define the price and conditions of a service supplied to the customer. When an account has active contracts, the system produces bills for it. If the account doesn't have active contracts, the system will not produce a bill for it. You can think of an account without active contracts as being "dormant", waiting for the day when the customer again starts service. If the customer never restarts, the account (along with its financial history) remains dormant forever.

Contracts

A contract is a contract (either formal or implied) between your organization and a customer. Every contract contains the price and conditions of a service supplied to a customer.

A contract is linked to an account. There is no limit to the number of contracts that may be linked to an account.

When Is A Contract Created?

A contract is created when the customer requests service (not when service commences). Typically, contracts are created in the pending state. Once the contract is activated, the bill segments can be generated for the contract.

Fastpath: For more information about starting service, refer to *The Big Picture Of Starting Service*. For more information about bill segments, refer to *Bill Details*.

Financial Transactions Are Linked To Contracts

Fastpath: For more information about how financial transactions are linked to contracts, refer to *The Financial Big Picture*.

When Is A Contract Expired?

A contract is expired when the customer requests service be stopped. At that time, the contract is transitioned to the `pending stop` state. Once the system transitions the contract into the `stopped` state, the billing process generates a final bill for the contract. When the customer pays the final bill, the system transitions the contract to the `closed` state

Fastpath: For more information about stopping service, refer to *The Big Picture Of Stopping Service*.

Setting Up Person Options

This section describes tables that must be set up before you can define persons.

Defining Person Identifier Types

When you create a person (customer), a person identifier type allows you to define the various types of identification associated with the person (customer). For example, driver's license number, taxpayer identification number, etc. The **Person Identifier Type** screen allows you to create, edit, and delete a person identifier type. You can access this screen using **Admin Menu, Person Identifier Type**.

You can quickly search for a person (customer) using a person identifier. In addition, person identifiers help prevent duplicate persons from being added to the database. Usually, the system warns a user before they add a new person when a person exists with the same identifier.

The person identifier types are optional. However, using installation options, you can mandate at least one identifier type for each person type.

Description of Page

The **Person Identifier Type** screen contains the following fields:

- **Person Identifier Type** - Used to specify the person identifier type.
- **Description** - Used to specify the description for the person identifier type.
- **Identifier Format** - Used to specify the algorithm that will be triggered when the details are saved in the **Person** screen. This algorithm will validate the value specified for the person identifier type.

Defining Person Relationship Types

It is possible to associate persons to other person. For example,

- You might want to define the subsidiaries of a parent corporation

- You might want to define spouses as separate persons and then link each person to another person

When you link a person to another person, you must define in what way the person is related to the other person by using a person relationship type code. These codes are defined using **Admin Menu, Person Relationship Type**.

Description of Page

Enter the following for each relationship type:

- Enter an easily recognizable **Relationship Type** code.
- Use **Description (Person1=>Person2)** to describe how the first person is related to the second person.
- Use **Description (Person2=>Person1)** to describe how the second person is related to the first person.

Note: Person1 versus Person 2. When you link persons together, you do it in respect of one of the persons (which we call Person 1). For example, if you want to link the subsidiaries to a parent company, you do this in respect of the parent company (i.e., you define the parent company's subsidiaries using the *Person - Persons* transaction.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_PER_REL_TYPE*.

Setting Up Statement Construct Options

This section describes tables that must be set up before a statement construct can be set up for a person to begin receiving financial statements.

Fastpath: For more information, refer to *The Big Picture of Complex Statements*.

Setting Up Statement Route Types

Statement route types define the method used to route statements to persons. To define a statement route type, open **Admin Menu, Statement Route Type**.

Description of Page

Enter a unique **Statement Route Type, Description** and **Statement Routing Method** for every statement route type.

Note: The values for Statement Routing Method are customizable using the Lookup table. This field name is STM_RTG_METH_FLG.

The next two fields control how statements that are routed using this route type are printed (both in batch and online). Refer to *Technical Implementation Of Batch Statement Production* for more information about producing statements in batch. Refer to *Technical Implementation Of Online Statement Production* for more information about online statement production.

- Use **Batch Control** to define the process that creates the flat file that is passed to your statement printing software. If you use an **Extract Algorithm** to construct the downloaded information, you can use the `STMDWLD` process.
- Use **Extract Algorithm** to define the plug-in component that constructs the "flat file records" that contain the information to be merged onto statements routed using this route type. This algorithm is called when a user requests an online image of a statement on *Statement - Main* and it may also be called by the batch statement extraction process defined above. Click *here* to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_STM_RTE_TY*.

Setting Up Account Options

This section describes tables that must be set up before an account can receive a bill.

Setting Up Account Management Groups

Users are informed that something requires their attention by entries that appear in To Do lists.

You can optionally use account management groups (AMG) to define the respective role to be assigned to To Do entries that are associated with an account and a given To Do type. For example, you can create an AMG called `Credit Risks` and assign this to accounts with suspect credit. Then, whenever an account-oriented To Do entry is created for such an account, it will be assigned a role based on the `Credit Risks` AMG. Refer to *Assigning A To Do Role* for more information..

Note: Account management groups are optional. You need only set up account management groups (and link them to accounts) if you wish to address specific To Do entries associated with specific accounts to specific roles.

Account management groups are defined using **Admin Menu, Account Management Group**.

Description of Page

Enter an easily recognizable **Account Management Group** code and **Description** for each account management group. Use the grid to define the **To Do Role** to be assigned to entries of a given **To Do Type** that are associated with accounts that reference the **Account Management Group**.

Note: Only To Do entries that are account-oriented take advantage of the roles defined for an account management group (because only accounts reference an account management group).

Where Used

Follow this link to view the tables that reference *CI_ACCT_MGMT_GR* in the data dictionary schema viewer.

Setting Up Account Relationship Codes

When you link a person to an account, you must define in what way the person is related to the account by using an account relationship code. These codes are defined using **Admin Menu, Account Relationship Type**.

Description of Page

Enter an easily recognizable **Relationship Type** and **Description** for each relationship type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ACCT_REL_TYP*.

Setting Up Alert Types

Account based alerts that appear in control central have an **Alert Type**. To define valid alert types, navigate to **Admin Menu, Alert Type.**

Description of Page

Enter an easily recognizable **Alert Type Code** and **Description** for each alert type. Specify the **Alert Days** to indicate the amount of time that alerts of this type will be effective by default. Specify a value of zero to indicate that alerts of this type will be effective indefinitely by default.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ALERT_TYPE*.

Setting Up Bill Messages

There are various informational and warning messages that may appear on an account's bills. Each message is identified with a bill message code. To define a bill message code, open **Admin Menu, Bill Message**.

Description of Page

Enter a unique **Message Code** and **Description** for every bill message.

The following attributes control how and where the bill message appears on the customer's bill:

Priority controls the order in which the message appears when multiple messages appear on a bill.

Note: The values for this field are customizable using the Lookup table. This field name is MSG_PRIORITY_FLG.

Insert Code controls whether a document should be inserted into the bill envelope when the bill message appears on a bill.

Message on Bill is the actual verbiage that appears on the customer's bill. If the message text is not static (e.g., field values need to be substituted into the body of the message), you can use the % *n* notation within the **Message on Bill** to cause field values to be substituted into a message. Refer to *Substituting Field Values Into A Bill Message* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_BILL_MSG*.

Setting Up Bill Route Types

Bill route types define the method used to route bills to accounts. To define a bill route type, open **Admin Menu, Bill Route Type**.

Description of Page

Enter a unique **Bill Route Type** and **Description** for every bill route type.

Bill Routing Method controls the type of information that may be defined when the respective **Bill Route Type** is selected on *Account - Person Information*. The following options are available:

- `Postal`. Use this method if the routing is via the postal service.
- `Fax`. Use this method if the routing is via fax.
- `Email`. Use this method if the routing is via email.

Note: The values for **Bill Routing Method** are customizable using the *Lookup* table. This field name is BILL_RTG_METH_FLG.

The next two fields control how bills that are routed using this method are *printed* (both in batch and online).

- Use **Batch Control** to define the background process that performs the actual download of the billing information. Refer to *Technical Implementation of Printing Bills In Batch* for more information about these processes.
- Use **Extract Algorithm** to define the algorithm that constructs the records that contain the information that appears on a printed bill. Refer to *Printing Bills* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_BILL_RT_TYPE*.

Setting Up Bill Cycles

Fastpath: Refer to *Defining Bill Cycles* for a description of how to set up bill cycles.

Setting Up Customer Classes

When you set up an account, you must assign it a customer class. The topics in this section describe the customer class control table.

Customer Class - Main

To set up customer classes, navigate to **Admin Menu, Customer Class - Main**.

Description of Page

Enter a unique **Customer Class** code and **Description** for every customer class.

Use **Collection Class** to define the collection class that defaults onto new accounts that belong to this customer class. An account's collection class may be subsequently modified if the account has special collection problems or needs.

Fastpath: For more information about the significance of collection class, refer to *Designing Your Collection Classes*.

Turn on **Business Activity Required** if contracts linked to accounts with this customer class require a Business Activity description to be entered.

Turn on **Open Item Accounting** if accounts belonging to this customer class are subject to open-item account. Refer to *Open Item Accounting* for a complete explanation of the significance of this switch.

Turn on **Non CIS Payment** if accounts belonging to this customer class are used for payments made to reduce non-CIS debt. For example, assume your company accepts payments for a county assessor and you don't want to set up a separate account for each person who pays their assessment bill. You should set up the following information to accept such payments:

- Create a new customer class called "Non CIS Customer".
- Create a contract type for each type of non-CIS payment that customers can make. Make sure to enter a distribution code on each contract type that references the appropriate revenue (or payable) account. Don't forget to indicate that each contract type is not billed.

Note: Payment Templates can be used for common types of non-CIS payment allocations. These templates are used to default the payment distribution and allow non-CIS payments to be directly allocated to specific distribution codes.

Fastpath: For more information about using Payment Templates to process non-CIS payments, refer to *Non-CIS Payments* Payments.

- Create an account to which you'll book such payments. Have this account reference the new customer class. We recommend creating a separate account for each contract type that you created in the previous step.
- Create and activate a contract for the new account(s).

When someone pays for non-CIS debt, the operator will add a payment for the above account. On the payment, the operator should record reference information in order to know exactly why the payment was made. Refer to *Payment Event - Main* for more information.

You must define a variety of business rules for every division in which a customer class has customers. For example, if you operate in both California and Nevada AND you have divisions for each state AND you have residential customers

in each state, you must define **Customer Class Controls** for each division. You do this on the *Customer Class - Controls* page. The grid that follows simply shows the divisions for which business rules have been set up.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CUST_CL*.

Customer Class - Bill Messages

When a customer class has bill messages, the system will sweep these messages onto bills created for accounts belonging to the customer class. Use this page to define a customer class's bill messages. Navigate to **Admin Menu, Customer Class, Bill Messages** tab to maintain this information.

Description of Page

Use the bill messagescollection to define **Bill Message** codes that should appear on bills that created for accounts that belong to a given customer class. For each message, also specify the **Start Date** and **End Date** when such a message should appear on the bill (leave **End Date** blank if the message should appear indefinitely).

Where Used

The system snaps customer class bill messages on a bill during bill completion. For more information about bill messages, refer to *The Source Of Bill Messages*.

Customer Class - Controls

You must define a variety of business rules for every division in which a customer class has customers. For example, if you operate in both California and Nevada AND you have divisions for each state AND you have financial services related customers in each state, you must define **Customer Class Controls** for each division in respect of the financial services related customer class. Open **Admin Menu, Customer Class, Controls** tab to maintain this information.

Description of Page

The **Customer Class Controls** scroll contains business rules governing accounts that belong to a **Division** and **Customer Class**. The following fields should be defined for each **Division**:

- Use **Days Till Bill Due** to define the number of days after the bill date that the customer's bill is due. If the due date is a weekend or company holiday, the system will move the due date forward to the next workday (using the workday calendar defined on the account's division).
- Specify the **Budget Plan** that defaults onto new accounts belonging to this customer class. Please note that an account's budget plan may be subsequently modified if the account has special budget processing needs. Refer to *Setting Up Budget Plans* for more information.
- Use **Min Credit Review Freq (Days)** to define the maximum number of days that can elapse between the reviews of an account's debt by the *account debt monitor*. Note, a value of zero (`0`) means that accounts in this customer class will be reviewed every day.
- Use **Credit Review Grace Days** to define the number of days after the bill due date that an account should be reviewed by the *account debt monitor*.
- Turn on the **Late Payment Charge** if customers in the class / division combination are eligible for late payment charges.
- Use **LPC Grace Days** to define the number of days after a bill's due date that a late payment charge will be generated (if the various LPC algorithms allow such - refer to *How Late Payment Charges Get Calculated* for the details). If the grace date falls on a weekend or holiday, the system moves the grace date to the next available workday (using the workday calendar defined on the account's division).

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

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).

- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

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

You can define algorithms for the following **System Events**:

System Event	Optional / Required	Description
Autopay Amount Over Limit	Optional	This algorithm is called to handle the situation when a system-initiated *automatic payment* is created that exceeds the customer's *maximum withdrawal limit*. Specifically, this algorithm is called when: - The account has a maximum withdrawal limit on their *automatic payment options* - The system attempts to create an automatic payment that exceeds this amount - The automatic payment algorithm that's plugged into the *installation record* has logic that invokes this algorithm when the above conditions are true If you do not plug-in this type of algorithm and the above situation is detected, the automatic payment will be created and no error will be issued. Refer to *How To Implement Maximum Withdrawal Limits* for more information. Click *here* to see the algorithm types available for this system event.
Bill Cancel	Optional	This algorithm provides the ability to include additional cancel logic when canceling online. Algorithms of this type can be called in two modes: D (Determine Bill Page Buttons) and X (Cancel Bill). Mode 'D' governs whether an action button to cancel the bill will appear on the Bill page and mode 'X' performs the actual cancellation logic. Click *here* to see the algorithm types available for this system event.
Bill Completion	Optional	When a bill for an account is completed, bill completion algorithms are called to do additional work. Refer to the description of the Complete button under *Bill Lifecycle* for a description of when this algorithm is called during the completion process. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Bill Eligibility	Optional	Algorithms for this plug-in spot are called when generating a bill in batch billing. It provides the ability to determine if an account is ineligible for billing and should therefore be skipped from further processing. If an eligibility algorithm is not used, a bill is created for any account in the open bill cycle and is later deleted by the billing process if it detects that there in no information linked to the bill. Click *here* to see the algorithm types available for this system event.
Bill Segment Freeze / Cancel	Optional	When a bill segment for an account in this customer class / division is frozen or canceled, an algorithm of this type may be called to do additional work. Refer to *Bill Segment Lifecycle* for more information about freezing and canceling bill segments. Click *here* to see the algorithm types available for this system event.
Bill Segment Information	Optional	When a customer class has an algorithm associated with this system event, the algorithm is called when a bill segment is generated. This algorithm generates the bill segment information string, which appears on the bill. It concatenates the fields and delimiters specified as parameters in the algorithm. You can maintain separate bill segment information string for each type of bill (i.e. customer class), such as individual bill, list bill, and SAP bill.
FT Freeze	Optional	When an FT is frozen, this algorithm is called to do additional work. For example, if you practice *Open Item Accounting*, you will need such an algorithm to handle the cancellation of match events when a financial transaction is canceled that appears on a match event. Refer to *How Are Match Events Cancelled?* for more information about cancellation. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Late Payment Charge Eligibility	Required if the customer class / division is eligible for late payment charges	This algorithm is called by the late payment process to determine eligibility for late payments.  **Caution: Just because an account's customer class allows late payment charges to be calculated doesn't mean the account's delinquent contracts will be levied late payment charges. In addition, a delinquent contract's contract type must reference a late payment charge algorithm. Refer to *Contract Type - Main* for more information about contract type late payment charge issues. Refer to *How Late Payment Charges Get Calculated* for more information about late payment charges in general.** **Note: Only One Algorithm**. Only one late payment charge eligibility algorithm may be defined for a customer class / division combination. Click *here* to see the algorithm types available for this system event.
Levy an NSF Charge	Optional	This algorithm is called when a payment is canceled with a cancellation reason that indicates an NSF. Refer to *NSF Cancellations* for more information about what happens when a payment is canceled due to non-sufficient funds. **Note: Only One Algorithm**. Only one algorithm to levy an NSF charge may be defined for a customer class / division combination. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Order Completion	Optional	When an *order* is completed for a customer linked to this customer class, this algorithm is called to do additional work (e.g., create a customer contact). You need only specify this type of algorithm if you require additional work to be performed when an order is completed for customers who belong to this customer class. Click *here* to see the algorithm types available for this system event.
Overpayment Distribution	Required	When a customer pays more than they owe, this algorithm is called to determine what to do with the excess funds. Refer to *Overpayment Segmentation* for a description on how to configure the system to handle your overpayment requirements. **Note: Only One Algorithm**. Only one overpayment distribution algorithm may be defined for a customer class / division combination. Click *here* to see the algorithm types available for this system event.
Override Due Date	Optional	An account's bill due date will be equal to the bill date plus its customer class' Days Till Due. If you need to *override* this method for accounts in a specific customer class, specify the appropriate algorithm here. **Note: Only One Algorithm**. Only one due date override algorithm may be defined for a customer class / division combination. Click *here* to see the algorithm types available for this system event.
Payment Cancellation	Optional	Algorithms of this type are called when a payment is canceled. Click *here* to see the algorithm types available for this system event.
Payment Distribution	Required	This algorithm is called to distribute a payment amongst an account's contracts. Refer to *Payment Distribution* for more information about how payment distribution works. **Note: Only One Algorithm**. Only one payment distribution algorithm may be defined for a customer class / division combination. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Payment Freeze	Optional	When a payment is frozen, this algorithm is called to do additional work. If you practice *Open Item Accounting*, you will need such an algorithm to link the payment's financial transactions to the match event that was originally created when the payment was distributed. Refer to *Payments and Match Events* for more information. Click *here* to see the algorithm types available for this system event.
Post Bill Completion	Optional	When a customer class has algorithms of this type, they are called after the completion of a bill for an account linked to this customer class. Refer to the description of the Complete button under *Bill Lifecycle* for a description of when this algorithm is called during the completion process. Click *here* to see the algorithm types available for this system event.
Pre Bill Completion	Optional	When a customer class has algorithms of this type, they are called immediately before completion starts for an account linked to this customer class. These algorithms have the potential of: • Deleting a bill. You might want a pre completion algorithm to delete a bill if a condition is detected that should inhibit the sending of a bill to a customer (e.g., the bill just contains information about recent payments). • Aborting the completion process and creating a bill exception. If the algorithm indicates this should be done, the bill is left in the `pending` state and a bill exception is created describing why completion was aborted. You might want a pre completion algorithm to do this if, for example, integrity checks detect there is something wrong with the account or its contracts. If the integrity check fails, the bill can be left in the `pending` state and a bill exception created describing why. Refer to the description of the Complete button under *Bill Lifecycle* for a description of when this algorithm is called during the completion process. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
Quote Completion	Optional	When a *quote* is completed for a customer linked to this customer class, this algorithm is called to do additional work (e.g., create a customer contact). You need only specify this type of algorithm if you require additional work to be performed when a quote is completed for customers who belong to this customer class. Click *here* to see the algorithm types available for this system event.
Write Off Method	Required if you allow users to write-off debt real time using the *write off transaction*	When a user presses the create button on the *write off transaction*, this algorithm is executed to write-off the selected debt. Refer to *The Ramifications of Write Offs in the General Ledger* for more information. Click *here* to see the algorithm types available for this system event.

Setting Up Collection Classes

Fastpath: Refer to *Setting Up Collection Classes* for a description of how to set up collection classes.

Account Identifier Type

When you create an account, an account identifier type allows you to define the various types of identification associated with the account. For example, bank account number, account name, International Bank Account Number (IBAN), etc. The **Account Identifier Type** screen allows you to create, edit, and delete an account identifier type. This screen consists of the following zones:

- *Search* on page 261

Search

The **Search** zone allows you to search for an account identifier type. This zone contains the following two sections:

- **Search Criteria** — The **Search Criteria** section contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Account Identifier Type	Used to specify the account identifier type.	No
Description	Used to specify the description of the account identifier type.	No

- **Search Results** — On clicking the **Search** button, the search results appear based on the specified search criteria. The **Search Results** section contains the following columns:

Column Name	Column Description
Account Identifier Type	Displays the account identifier type.
Description	Displays the description of the account identifier type.

Column Name	Column Description
Validation Algorithm	Indicates the algorithm that will be triggered when the details are saved in the **Account** screen. This algorithm validates the value specified for the account identifier type.
Edit	On clicking the **Edit** ( ) icon, the **Edit Account Identifier Type** screen appears where you can edit the details of the account identifier type.
Delete	On clicking the **Delete** ( ) icon, you can delete the account identifier type. **Note:** You can delete an account identifier type only if it is not yet used for any account or division.

You can create an account identifier type by clicking the **Add** link in the upper right corner of this zone.

Related Topics

For more information on...	See...
How to search for an account identifier type	*Searching for an Account Identifier Type* on page 262
How to create an account identifier type	*Creating an Account Identifier Type* on page 263
How to edit an account identifier type	*Editing an Account Identifier Type* on page 264
How to delete an account identifier type	*Deleting an Account Identifier Type* on page 265

Searching for an Account Identifier Type

Procedure

To search for an account identifier type:

1. Click the **Menu** link in the **Actions/Navigation** area.

   A list appears.
2. Select the **Admin Menu** option from the list.
3. From the **Admin Menu**, select **A** and then click **Account Identifier Type**.

   The **Account Identifier Type** screen appears.
4. Enter the search criteria in the **Search** zone.

   Note: ORMB search engine supports wildcard search, where you can substitute the percentage (%) symbol as a stand in for any word or letter in a search criteria. You can use the '%' wildcard character in all input fields except the date and ID fields. The '%' wildcard character is suffixed automatically at the end of the partial search criteria. Therefore, you may or may not specify the wildcard character at the end of the partial search criteria. However, you have to prefix the wildcard character manually wherever required.
5. Click **Search**.

   The search results appear.

Related Topics

For more information on...	See...
Account Identifier Type screen	*Account Identifier Type* on page 261
Search zone	*Search* on page 261

Creating an Account Identifier Type

Prerequisites

To create an account identifier type, you should have:

- Validation algorithm defined using the C1_DIVACCNBR or C1_SYSACCNBR algorithm type (in case you want to it)

Procedure

To create an account identifier type:

1. Click the **Menu** link in the **Actions/Navigation** area.

   A list appears.
2. Select the **Admin Menu** option from the list.
3. From the **Admin Menu**, select **A** and then click the **Account Identifier Type**.

   The **Account Identifier Type** screen appears.
4. Click the **Add** link in the upper right corner of the **Search** zone.

   The **Add Account Identifier Type** screen appears. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Account Identifier Type	Used to specify the account identifier type.	Yes
Description	Used to specify the description for the account identifier type.	Yes
Validation Algorithm	Used to specify the algorithm that will be triggered when the details are saved in the **Account** screen. This algorithm validates the value specified for the account identifier type. The following algorithm types are shipped with ORMB: • **C1_DIVACCNBR** — Used when you do not want accounts with the same account identifier type and account identifier combination in the division. • **C1_SYSACCNBR** — Used when you do not want accounts with the same account identifier type and account identifier combination across divisions.	No

5. Enter the required details.

   Note: You can search for an algorithm by clicking the **Search** () icon corresponding to the field.
6. Click **Save**.

   The account identifier type is created.

Related Topics

For more information on...	See...
Account Identifier Type screen	*Account Identifier Type* on page 261
Search zone	*Search* on page 261

Editing an Account Identifier Type

Prerequisites

To edit an account identifier type, you should have:

- Validation algorithm defined using the C1_DIVACCNBR or C1_SYSACCNBR algorithm type (in case you want to it)

Procedure

To edit an account identifier type:

1. Search for the account identifier type in the **Account Identifier Type** screen.
2. In the **Search Results** section, click the **Edit** (✎) icon in the **Edit** column corresponding to the account identifier type whose details you want to edit.

   The **Edit Account Identifier Type** screen appears. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Account Identifier Type	Displays the account identifier type.	Yes
Description	Used to modify the description of the account identifier type.	Yes
Validation Algorithm	Used to specify the algorithm that will be triggered when the details are saved in the **Account** screen. This algorithm validates the value specified for the account identifier type. The following algorithm types are shipped with ORMB: • **C1_DIVACCNBR** — Used when you do not want accounts with the same account identifier type and account identifier combination in the division. • **C1_SYSACCNBR** — Used when you do not want accounts with the same account identifier type and account identifier combination across divisions.	No

3. Modify the required details.

   Note: You can search for an algorithm by clicking the **Search** (🔍) icon corresponding to the field.

4. Click **Save**.

   The changes made to the account identifier type are saved.

Related Topics

For more information on...	See...
How to search for an account identifier type	*Searching for an Account Identifier Type* on page 262
Account Identifier Type screen	*Account Identifier Type* on page 261
Search zone	*Search* on page 261

Deleting an Account Identifier Type

Procedure

To delete an account identifier type:

1. Search for the account identifier type in the **Account Identifier Type** screen.
2. In the **Search Results** section, click the **Delete** ( ) icon in the **Delete** column corresponding to the account identifier type that you want to delete.

   A message appears confirming whether you want to delete the account identifier type.

   Note: You can delete an account identifier type only if it is not yet used for any account or division.
3. Click **OK**.

   The account identifier type is deleted.

Related Topics

For more information on...	See...
How to search for an account identifier type	*Searching for an Account Identifier Type* on page 262
Account Identifier Type screen	*Account Identifier Type* on page 261
Search zone	*Search* on page 261

Setting Up Customer Contact Options

This section describes tables that must be set up before you can define customer contacts.

Fastpath: Refer to *The Big Picture Of Customer Contacts* for more information about customer contacts.

Setting Up Letter Templates

You can set up a customer contact type to generate a form letter whenever a customer contact of this type is added. In fact, this is the only way to generate a letter in the system.

Fastpath: Refer to *Printing Letters* for more information about how letters are produced.

Every customer contact that causes a letter to be sent must reference a unique letter template. To define a letter template, open **Admin Menu, Letter Template**.

Note: Doc 1 users. If you use the Doc 1 software to produce letters, there will be a template in the Doc 1 software associated with each letter. The name of the Doc 1 template must be the same as the code associated with the letter template set up in the system.

Description of Page

The following fields are required for each letter template:

- **Letter Template** is the unique identifier of the letter template.
- Use **Description** to enter a brief description of the letter.
- Turn on **Special Extract** if this type of letter should only be created via a system generated event such as a collection letter. Turning on this switch is what prevents a user from adding a customer contact that references this type of letter template (because you don't want a user to be able to request a letter associated with a system generate event by adding a customer contact, rather, they must execute the appropriate process and it will generate the customer contact).
- The next two fields control how letters of this type are printed (both in batch and online). Refer to *Technical Implementation Of Batch Letter Production* for more information about producing letters in batch. Refer to *Technical Implementation Of Online Letter Production* for more information about online letter production.
- Use **Batch Control** to define the process that creates the flat file that is passed to your letter printing software. If you use an **Extract Algorithm** to construct the downloaded information, you can use the LTRPRT process.
- Use **Extract Algorithm** to define the plug-in component that constructs the "flat file records" that contain the information to be merged onto letters of this type. This algorithm is called when a user requests an online image of a letter on *Customer Contact - Main* and it may also be called by the batch letter extraction process defined above. Click *here* to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_LETTER_TMPL*.

Setting Up Customer Contact Classes

Every customer contact record has a contact type that classifies the record for reporting purposes. And every contact type, in turn, references a customer contact "class". The class categorizes customer contacts into larger groupings for reporting purposes.

Open **Admin Menu, Customer Contact Class** to define your customer contact classes.

Description of Page

Enter a unique **Contact Class** and **Description** for each customer contact class.

After you have created your customer contact classes, you'll be ready to setup your *customer contact types*.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CC_CL*.

Setting Up Customer Contact Types

Every customer contact record has a contact type that controls the behavior of the customer contact.

Fastpath: Refer to *The Big Picture Of Customer Contacts* for more information about customer contacts.

Open **Admin Menu, Customer Contact Type** to define your customer contact types.

Description of Page

Every customer contact type is identified by a unique combination of **Contact Class** and **Contact Type**.

Enter a brief **Description** of the customer contact type.

Only specify a **Contact Shorthand** if customer contacts of this type can be added in the *Customer Contact Zone*. The value you specify in this field is what the user selects to add a customer contact in this zone.

Use **Contact Action** if something should be triggered when customer contacts of this type are added. The only valid value in this release is `Send Letter`. If you select this option, you must also specify a **Letter Template**. Refer to *Printing Letters* for more information about how letters are produced.

Use the **Customer Contact Type Characteristics** collection to define characteristics that can be defined for contacts of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on customer contacts of a given type. Turn on the **Default** switch to default the **Characteristic Type** when customer contacts of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CC_TYPE*.

Setting Up Contract Options

This section describes tables that must be set up before you can define contracts.

Setting Up Standard Industry Codes (SIC)

A contract for financial services should reference a standard industry code (SIC). This code is used to categorize contracts for reporting purposes. To define a SIC, open **Admin Menu, SIC Code**.

Description of Page

Enter a unique **SIC Code** and **Description** for the SIC.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SIC*.

Setting Up Tax Exempt Types

Your rates will probably have provisions for the calculation of taxes of one type or another. Frequently you will have customers who are completely or partially exempt from these taxes. The contracts for these customers will need to have tax exemption information in order for them to be billed properly. Tax Exempt Type is used to define the precise nature of the applicable exemption. To define the Tax Exempt Types you will use, open **Admin Menu, Tax Exempt Type.**

Description of Page

Enter a unique **Tax Exempt Type** and **Description** for each type of tax exemption.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_TAX_EX_TYPE*.

Setting Up Contract Quantity Types

You may have customers whose contracts have contractual consumption limits. The contracts for these customers must have information regarding this quantity in order to be billed properly. Contract Quantity Type is used to precisely define the nature of the quantity. To define the Contract Quantity Types, open **Admin Menu, Contract Quantity Type.**

Description of Page

Enter a unique **Contract Quantity Type** and **Description** for each type of contract quantity.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CONT_QTY_TYP*.

Contract Type Controls Everything

Every contract references a contract type. The contract type controls all aspects of a contract's behavior including how service is started, how bills are created, how its financial transactions are booked in the general ledger, and much more. We don't explain how to set up contract types in this section because it's only after you have set up all of the control tables in this manual that you'll be able to finally define your contract types.

Fastpath: For more information about contract types, refer to *Defining Contract Types*.

Financial Controls

Fastpath: There are also a number of control tables that must be set up to control the bills, payments, and adjustments that are linked to a contract. For more information about these tables, please refer to *Defining Financial Transaction Options*.

Setting Up Order Options

This section describes tables that must be set up before orders can be used to start service.

Fastpath: For more information, refer to *The Big Picture of Campaigns, Packages and Orders*.

Setting Up Column References

A column reference must be created for each miscellaneous field that's captured on an order that doesn't reside in a characteristic. Refer to *Determine The Properties Of Every Miscellaneous Field* for more information.

Open **Admin Menu, Column Reference** to define your column references.

Description of Page

Enter an easily recognizable **Column Reference** code and **Description** for each column reference.

Specify the **FK Reference** to use if this column reference uses field values from another table. Use **Long Description** to describe the data that fields using this column reference capture.

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

- Specify the **System Event** with which the algorithm is associated. You can define algorithms for the following system events: `Post when order completed`, `Retrieve current value`, `Validate field value`. Refer to *Extract Column References* for a description of these events.
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_COL_REF*.

Setting Up Order Cancellation Reasons

An order cancellation reason must be supplied when an order is cancelled. Open **Admin Menu, Order Cancel Reason** to define your reason codes.

Description of Page

Enter an easily recognizable **Order Cancel Reason** and **Description** for each order cancellation reason.

Where Used

Cancellation reasons are used when an *order is canceled*.

Setting Up Order Hold Reasons

An order hold reason must be supplied when an order is held. Open **Admin Menu, Order Hold Reason** to define your reason codes.

Description of Page

Enter an easily recognizable **Order Hold Reason** and **Description** for each order hold reason.

Where Used

Hold reasons are used when an *order is held*.

Setting Up Order Feature Configurations

Defining a *feature configuration* with a feature type of `Order Configuration` can increase performance of the Order page when campaigns have a large number of packages or criteria. Open **Admin Menu, Feature Configuration** to define a configuration for the feature type `Order Configuration`.

Note: Only one. The system expects only one `order configuration` feature configuration to be defined.

Description of Page

The following points describe the various **Option Types** that may be defined:

- `Eligibility Tree - Suppress Error Packages node.` Select this option type and define a value if you would like the *Order Eligibility Tree* to suppress the node that contains packages with errors in their eligibility criteria. This is an optional setting. If the option type is not defined, the error packages node is displayed, if applicable.
- `Eligibility Tree - Suppress Ineligible Packages node.` Select this option type and define a value if you would like the *Order Eligibility Tree* to suppress the node that contains packages that are not applicable to the customer based on the eligibility criteria. This is an optional setting. If the option type is not defined, the ineligible packages node is displayed, if applicable.
- `Eligibility Tree - Suppress Other Campaigns node.` Select this option type and define a value if you would like the *Order Eligibility Tree* to suppress the node that contains other eligible campaigns. This is an optional setting. If the option type is not defined, the other campaigns node is displayed, if applicable.

Chapter 5

Defining Credit & Collections Options

Topics:

Note: Collecting on unpaid balances. The functionality described in this section is meant to handle the collection of unpaid balances. If your organization practices *open-item accounting* and collects on unpaid bills, you will not use this functionality. Rather, you will use the functionality described under *Defining Overdue Processing Options*.

The system periodically monitors how much your customers owe to ensure they haven't violated your collection criteria. When a violation is detected, the system generates the appropriate responses (e.g., letters, disconnect notices, collection agency referrals, and eventually write off). This section describes how to set up the tables that control your credit & collections processing.

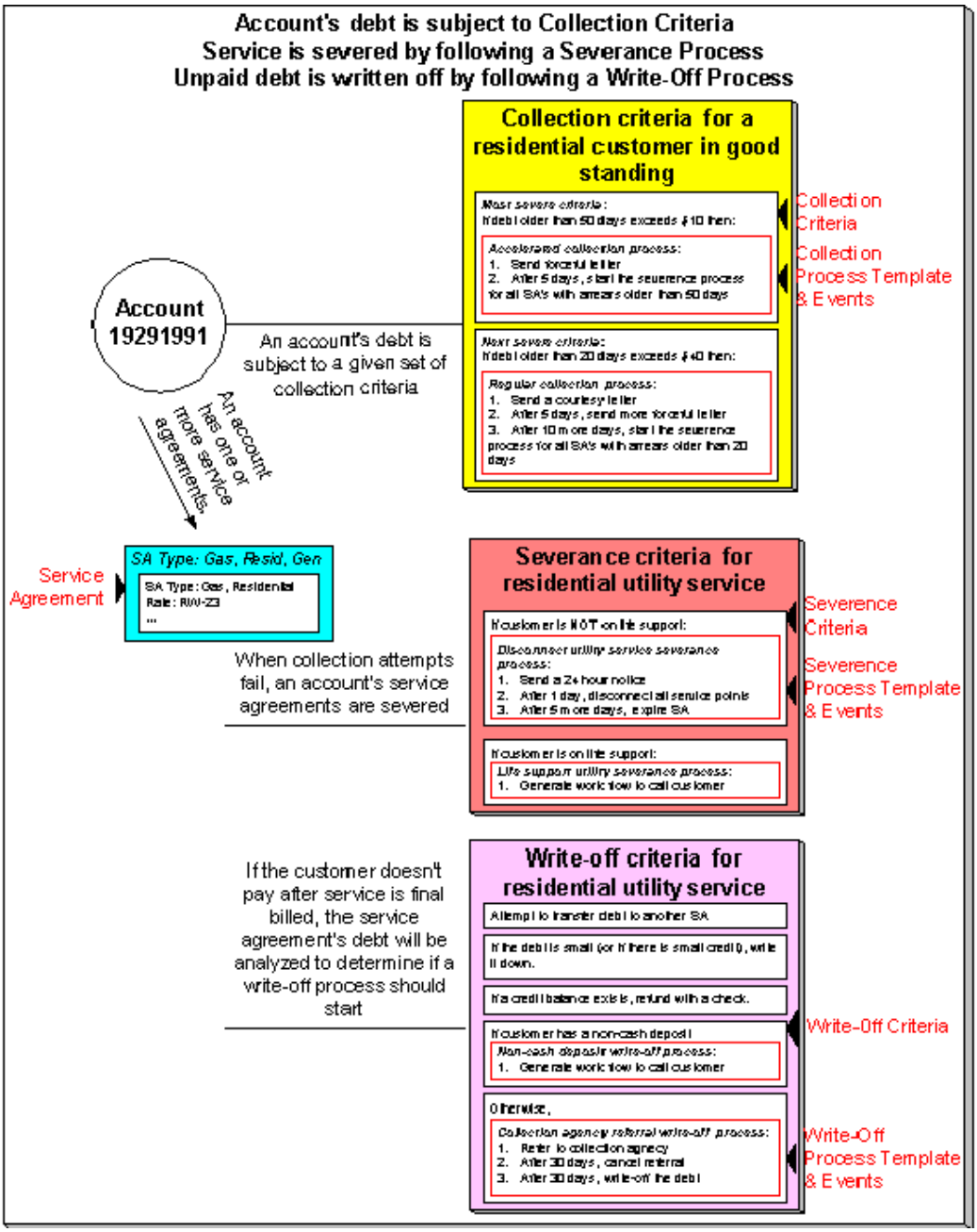
Caution: Setting up the credit & collections control tables is as challenging as your organization's collection rules. If you have simple rules then your setup process will be straightforward. If your collection rules are complicated (e.g., they differ based on the type of customer, the type of debt, the age of debt, the type of service, etc.), then your setup process will be more challenging.

The Big Picture Of Credit & Collections (C&C)

This section provides an overview of important C&C concepts with which you should be familiar before you set up your C&C control tables.

Collection Criteria vs. Write Off Criteria

The following diagram introduces important concepts related to the C&C processes:



There are many important concepts illustrated above:

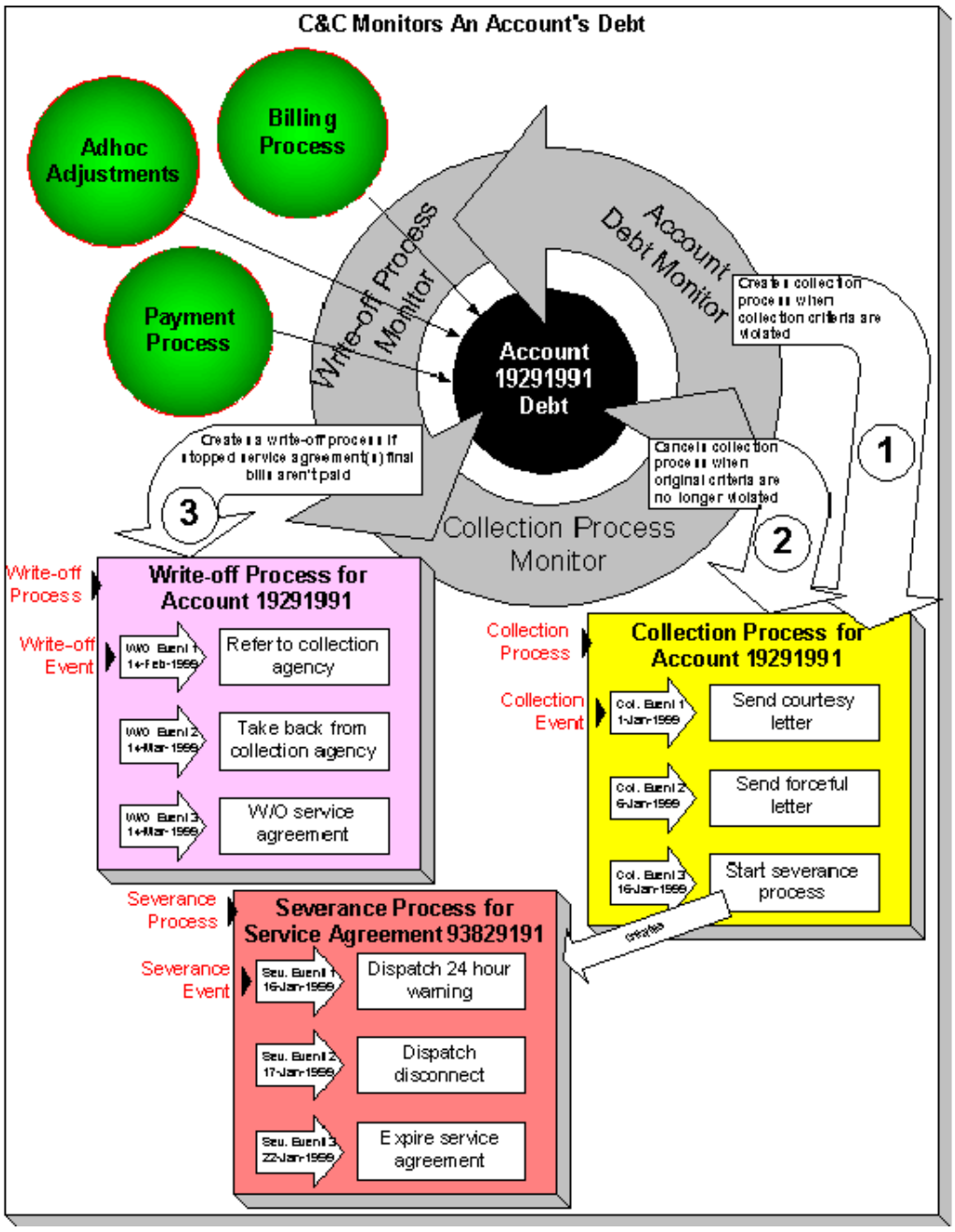
An account's debt comes from its contracts	An account's debt is managed at the contract level, i.e., the system keeps track of how much a customer owes in respect of each contract. In order to determine an account's balance, the system must add up the debt on each of the account's contracts.
Collection criteria define intolerable debt	Collection criteria are control data that define intolerable debt. Most criteria are defined using a combination of number of days in arrears and a dollar amount.
Collection criteria may be compared to an account's total debt or to subsets of debt	If your organization has simple collection procedures, you will probably target collection criteria at an account's total debt. However, you have the option of segregating an account's debt into debt classes and targeting the collection criteria at each class. For more information about debt classes, see *Different Collection Criteria For Different Customers And Different Debt* .
Collection criteria also define what to do when the level of intolerable debt is exceeded	When you define collection criteria, you also define how the system should respond if an account violates your criteria. These collection events are defined in respect of a "collection process template".
There are usually several collection events that take place when criteria are violated	A collection process template usually has several collection events. Each event is meant to prod the customer to pay. The initial collection events are typically letters.
After a contract is severed, it will be final billed	When the last active contract linked to an account is stopped, the system changes the account's bill cycle to bill that evening. If only one of many contract's is stopped, the contract will only be final billed as per the account's original bill cycle schedule.
If a customer doesn't pay their final bill, the account's debt will be analyzed to determine if the system can reduce the debt to zero using a variety of mechanisms	The system will look at an account's finaled debt on its next scheduled credit review date (typically a few days after the bill's due date). The system will attempt to reduce the contract's debt to zero using all of the following methods: • If the account has active contracts, it will transfer the finaled debt to an active contract. • If the debt or credit amount on the contract is small, the system will generate an adjustment to 'write it down' (or up in the case of a small credit). • If the contract has a large credit amount, the system will generate an A/P adjustment (resulting in a check being sent to the customer).

If a customer's finaled debt cannot be reduced via any of the previous methods, the system creates a write-off process	A write-off process contains one or more write-off events. These events can generate a letter, send a To Do entry to a CSR, send a referral to a collection agency, etc. When you set up the system, you define the type of write-off process to use for every collection class / write-off debt class combination. In addition, you can also indicate when the type of write-off process should differ depending on some attribute of the customer. For example, you may have a different write-off process if the customer has a non-cash deposit.
The last write-off event typically causes the debt to be written off	Ultimately, if the write-off events fail, the debt will have to be written off. When debt is written-off, the system creates a write-off contract and transfers the outstanding debt to it. This means the debt stays with the account for life and will have to be paid off if the customer ever returns.

Note: Checkpoint. At this point, you should be familiar with the concept that an account's debt is compared to user-defined collection criteria. If the account violates the criteria, a series of events will ensue that prod the customer to pay. If the customer doesn't respond, every contract in arrears will be severed (i.e., disconnected). If lack of service doesn't inspire payment, the contract will be expired and a write-off process will be created to manage the write-off activities.

The C&C Monitors

Your collection and write-off criteria described in the previous section exist to support the processes that manage the collection activities. The following diagram illustrates, at a high level, the major processes that manage the collection of overdue debt:



There are many important concepts illustrated above:

Bills, payments and adjustments affect an account's debt	An account's debt is the accumulation of all bills, payments and adjustments.
The Account Debt Monitor creates a collection process when an account violates collection criteria	Periodically, a background process referred to as the Account Debt Monitor (ADM and ADM2) determines if an account's debt violates your collection criteria. If so, a collection process is created using the violated criteria's collection process template. Refer to *When Is An Account's Debt Monitored?* for a description of when an account's debt is compared against collection criteria.

A collection process contains one or more collection events	The collection process contains a series of collection events. These events correspond with the collection event types associated with the collection process template. The initial collection events are typically letters.
The Collection Process Monitor cancels a collection process when warranted	The Collection Process Monitor cancels a collection process when its contracts satisfy your cancellation criteria (e.g., when the contracts have less than $10 of debt older than 20 days). Refer to *How Are Collection Processes Cancelled* for more information about the cancellation process.
The Write-Off Monitor creates a write off process to collect stopped, unpaid debt	The Write-Off Monitor reviews stopped and reactivated contracts after their closing bill's due date (plus grace period). The Write-Off Monitor attempts to reduce the contract's debt to zero using all of the following methods: • If the account has active contracts, it will transfer the finaled debt to an active contract. • If the debt or credit amount on the contract is small, the system will generate an adjustment to 'write it down' (or up in the case of a small credit). • If the contract has a large credit amount, the system will generate an A/P adjustment (resulting in a check being sent to the customer). If the system is unsuccessful in reducing the account's debt to zero, a write-off process will be created using the appropriate write-off process template. Refer to *The Big Picture Of Write Off Processing* for more information about the write-off process.
A write-off process contains one or more write-off events	The write-off process contains a series of write-off events. These events correspond with the write-off event types associated with the write-off process template. The initial write-off events are typically collection agency referrals and/or letters. If payment is not received as a result of such efforts, the last write-off event typically writes off the customer's debt.
The system cancels a write-off process when warranted	The system cancels a write-off process when its contracts no longer have debt (i.e., they become closed).
Another write-off process will be created if a closed contract ever reactivates	If a contract becomes reactivated (e.g., because the final payment bounces), the contract will be processed by the Write-Off Monitor and the whole write-off process starts again.

Note: Checkpoint. At this point, you should be familiar with the concept that a collection process will be created for an account that violates collection criteria. The collection process consists of a series of events that typically generate letters and / or To Do entries. If the customer doesn't pay the final bill, a write-off process will be created for each type of unpaid debt. The write-off process consists of a series of events that ultimately result in the write-off of the customer's debt. When debt is written-off, the system creates a write-off contract and transfers the outstanding debt to it. This means the debt stays with the account for life (because the write-off contract is linked to the account) and will have to be paid off if the customer ever returns.

The Big Picture Of Collection Processes

The topics in this section describe how collection processes are created and cancelled.

Fastpath: For more information refer to *The Lifecycle Of A Collection Process And Its Events* .

How Does The Account Debt Monitor Work?

This section describes how the Account Debt Monitor uses your collection criteria and collection process templates to collect overdue debt.

Different Collection Criteria For Different Customers And Different Debt

Consider the following:

- You probably have different collection criteria for different jurisdictions (i.e., Divisions). For example, if you have customers in different states / provinces, you may have different regulator-imposed criteria applied to each state's debt. You differentiate your debt in respect of the collection process via the **division code on each customer's account**.
- You probably have different collection criteria for different customer segments. For example, customers with large bills probably have strict criteria, whereas you're probably more lenient with small customers (or vice versa). You differentiate your customers in respect of the collection process via a **collection class code on the customers' accounts**. An account's initial collection class is defaulted from its customer class. You may override an account's collection class at will.
- You probably have different collection criteria for different classes of debt. You differentiate your debt in respect of the collection process via a **debt class code on the customers' contracts** (note: the debt class is actually defined on the contract's contract type).
- You will have different criteria for every currency in which you work because the monitoring process always compares a customer's debt against some value and this value must be denominated in the customer's currency. A customer's currency is defined using a **invoice currency on the account**.

Given the above, you should understand that different collection criteria will exist for every combination of division, collection class, debt class, and currency code. If you're confused, consider the following matrix (where we assume you have a single currency and division and therefore avoid the third and fourth dimensions):

Account's Collection Class **Contract's Debt Class**	Commercial Customer	Residential Customer
Regulated	N/A - there is no regulated, commercial customer debt.	Highest Priority: If > $5 in arrears by more than 50 days, create the accelerated collection process for residential customers. Lower Priority: If > $25 in arrears by more than 25 days, create the courtesy reminder collection process for residential customers.

Unregulated	Highest Priority: If > $10 in arrears by more than 50 days, create the accelerated collection process for commercial customers. Lower Priority: If > $1000 in arrears by more than 25 days, create the normal collection process for commercial customers.	Highest Priority: If > $10 in arrears by more than 25 days, create the normal collection process for residential customers.
Charitable Contribution	Highest Priority: If > $10 in arrears by more than 50 days, create the charitable collection process.	Highest Priority: If > $10 in arrears by more than 50 days, create the charitable collection process.

Also, notice that there can be multiple criteria for each cell in the matrix. What differentiates one collection criteria from another is its priority. The higher priority criteria will be compared first. If the debt meets the criteria, the collection process is initiated and no further comparisons are performed.

Fastpath: For more information about maintaining this matrix, refer to *Setting Up Collection Class Controls*. For more information about how the system handles an element in this matrix that has multiple criteria, see *How Is An Account's Debt Monitored?*.

Override Conditions

Caution: Your credit & collection requirements may not require any overrides and therefore this section may not be relevant for your organization.

The matrix presented in the previous section showed:

- You can have different collection criteria for different categories of debt and customers.
- When a collection criteria is violated, the system generates a specific collection process.

This works great for many organizations, but if your organization has other factors that affect either the collection criteria OR the collection process that is initiated when the criteria is violated, you may need to use override collection criteria.

Fastpath: Refer to *Designing Your Collection Class Control Overrides* for more information.

This section describes how and when the Account Debt Monitor analyzes an account's debt.

When Is An Account's Debt Monitored?

The account debt monitor (ADM) analyzes an account's debt at least every X days, where X is defined on the *customer class control* associated with the account's customer class and division (in the field Min Credit Review Freq (Days)).

In addition, an account's debt will also be monitored as follows:

- The ADM looks at an account's debt X days after an account's bill due date (X is defined on the account's customer class in the field Collection Grace Days).
- The ADM looks at an account's debt after a payment is canceled when the cancellation reason indicates NSF (non-sufficient funds).

- The ADM looks at an account's debt after a payment arrangement is broken (assuming you use the base package's break payment arrangement plug-in). Refer to *Monitoring Payment Arrangements* for more information.
- The ADM looks at an account's debt after a promise to pay is broken. Refer to *The Promise To Pay Monitor* for more information.

How Is An Account's Debt Monitored?

Assume the following collection control matrix exists for your organization:

Account's Collection Class **Contract's Debt Class**	Commercial Customer	Residential Customer
Regulated	N/A - there is no regulated, large customer debt	Highest Priority: If > $5 in arrears by more than 50 days, create the accelerated collection process for residential customers. Lower Priority: If > $25 in arrears by more than 25 days, create the courtesy reminder collection process for residential customers.
Unregulated	Highest Priority: If > $10 in arrears by more than 50 days, create the accelerated collection process for commercial customers. Lower Priority: If > $1000 in arrears by more than 25 days, create the normal collection process for commercial customers.	Highest Priority: If > $10 in arrears by more than 25 days, create the normal collection process for residential customers.

This matrix contains the information used by the Account Debt Monitor.

Fastpath: For more information about the information in this matrix, refer to *Different Collection Criteria For Different Customers And Different Debt*.

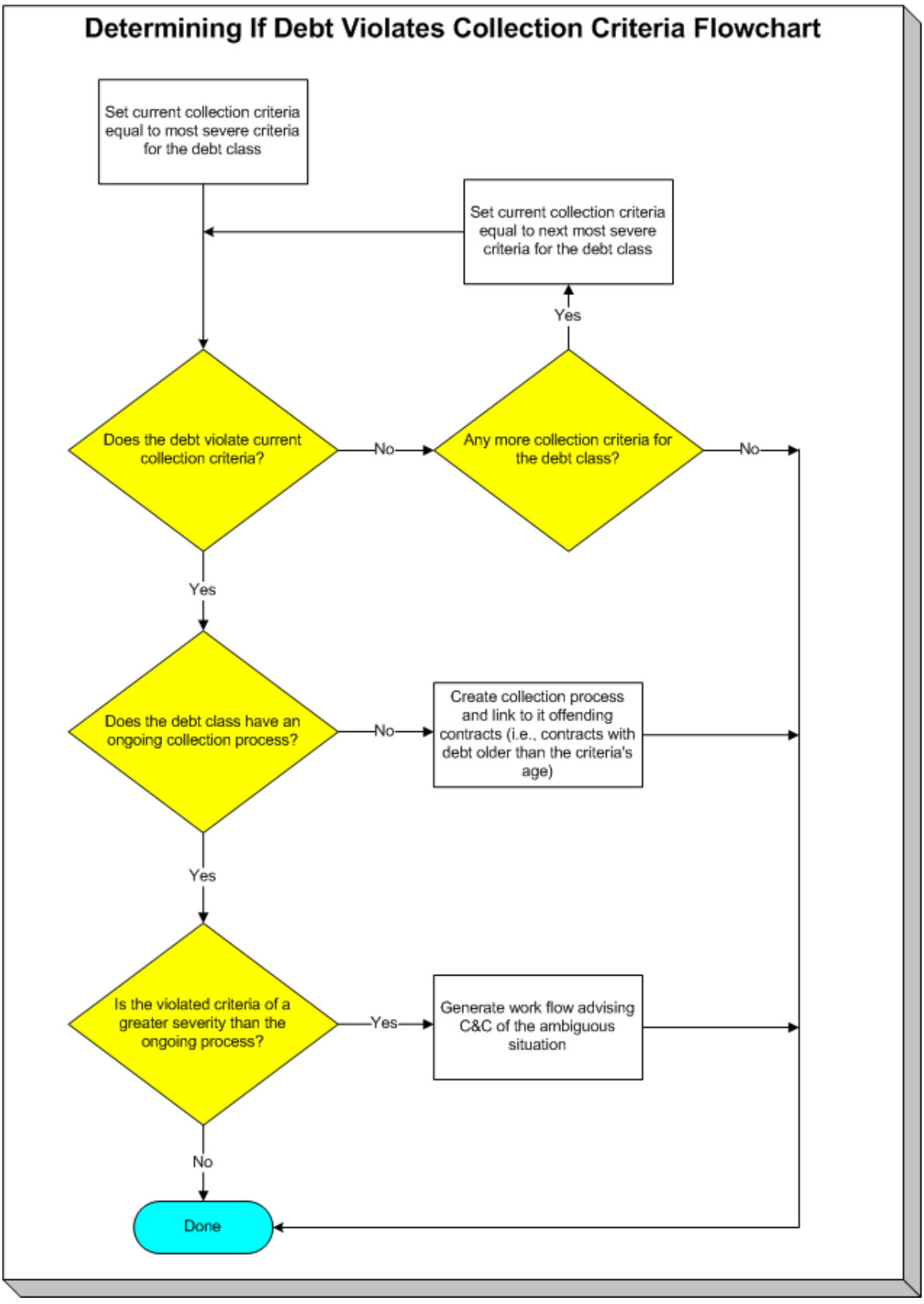
This matrix can be overwhelming when viewed as a whole. So let's consider how to use it for a specific account's debt and things will become clearer.

First, because an account belongs to a unique collection class, we only have to worry about a single column in the matrix when monitoring an account's debt.

Next, we accumulate the total amount of aged debt for each unique debt class associated with the account's contracts.

Next, we subject the accumulated aged debt to the override aged debt algorithm (plugged in on the debt class). This algorithm can cause aged debt to be reduced. This is an optional algorithm and is only used if you set up promise to pay for customers. Refer to *How Promise To Pay Affect The ADM* for more information.

Next, we determine if the debt for the debt class violates the collection criteria in the respective matrix element. If so, we kick off a collection process and link the offending contracts to it. The logic associated with the determination of whether to kick off a collection process is rather sophisticated. The following flowchart explains the exact details.



Note: Multiple collection processes may be kicked off. It's important to be aware that if an account's contracts reference multiple debt classes, a collection process will be started for each offending debt class.

Note: One collection process per debt class. A given debt class for an account may only have one ongoing collection process at any point in time.

What Happens When A Collection Process Is Started?

When you define collection criteria, you must define the collection process template to use if the criteria are violated. The system uses this template to create the account-specific collection process.

Every contract that is part of the offending debt class that has debt older than X days will be linked to the collection process (where X is the debt age on the collection criteria).

Also linked to the collection process will be one or more collection events. These events are typically a series of letters meant to prod the customer (you can also create an event that sends a To Do entry to a user to highlight the offensive debt). You define exactly which letters are generated and when they are generated when you set up the events on your collection process templates.

It's important to note that all of the collection events will be created when the collection process is created. Each of these collection events contains a trigger date. The trigger date of the first event(s) will typically be the current date. The trigger date of the other events will be in the future. Refer to *Calendar vs Work Days* for information that describes how the trigger date is set.

A separate process, Activate Collection Events, is responsible for activating collection events whose date is on or before the current date. Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do entry to a user, etc.)

If adequate payments / credits are recorded in the system, the collection process will be cancelled.

Fastpath: For more information about collection process templates, see *Setting Up Collection Process Templates*. For more information about collection events, see *The Big Picture Of Collection Events*. For more information about how a collection process is cancelled, see *How Are Collection Processes Cancelled*.

Experimenting With Alternative Collection Process Templates

The system allows you to determine the efficacy of proposed collection process templates using a small subset of customers before implementing the templates on the entire customer base. We use the term "champion / challenger" to reference this functionality.

We'll use an example to explain. Let's assume your prevailing collection process template for residential customers starts with a "gentle reminder" letter followed 10 days later by a letter threatening collection agency referral if payment is not received. You may want to experiment with the impact of a change to this template. For example, you may want to change the "gentle reminder" to something more assertive and follow this up 5 days later with an even sterner warning. You can use the "champion / challenger" functionality to perform this experiment.

The following points describe how to implement "champion / challenger" functionality:

- Set up a "challenger" collection process template for each template that you want to experiment with.
- Insert a new **Champion/Challenger** option on the `Collection Processing` *Feature Configuration* for every champion template. Each option's value defines:
  - the "champion" collection process template code
  - the "challenger" collection process template code
  - the percentage of the time the system should use the "challenger" template

- Keep in mind that you can only experiment with one challenger template per champion template. For example, let's assume you have two prevailing collection process templates - one for residential customers and another for commercial customers. You can experiment with different challenger templates for the residential and commercial templates. However, you cannot experiment with two different challenger templates for the residential champion template (i.e., a champion template can have 0 or 1 challenger template).

After setting up the above, the *Account Debt Monitor* will use the challenger template X% of the time rather than the champion template.

How Are Collection Processes Cancelled?

A collection process may be cancelled via the mechanisms described in this section.

The Collection Process Monitor Can Cancel A Collection Process

The Collection Process Monitor (CPM) is a background process that reviews a collection process when the debt associated with one of its contracts is reduced. Financial events that can cause contract debt to be reduced are:

- The cancellation of a bill segment.
- The creation of a payment segment.
- The creation of an adjustment that credits a contract.

The review performed by the CPM occurs as follows:

- **Debt class cancel criteria.** In general, the sum of all debt associated with the collection process's debt class must be less than a given threshold amount for a collection process to be cancelled. If so, the collection process is cancelled.
- Please be aware that, if a *Promise To Pay* exists for the account and debt class, the customer's debt will be temporarily reduced by the amount of the promise to pay's scheduled payments before it is compared to the threshold amount. Please be aware that this temporary reduction will only occur if you have plugged in the appropriate promise to pay debt reduction algorithm on the debt class.

Note: The above logic is not "hard coded". The CPM calls the *Collection Process Cancel Criteria Algorithm* defined on the debt class that is associated with the collection process. This algorithm will cancel a collection process if the sum of ALL contracts in the debt class have debt less than a given threshold amount. However, because it's an algorithm, you can introduce whatever cancellation criteria you please.

- **Contract cancel criteria.** You can *optionally* introduce a special quirk to the cancellation logic. This quirk is a bit difficult to understand. To understand it, you should recall:
- - All contracts that are in arrears in a given debt class are linked to the collection process.
- To "remove" contracts from a collection process when they no longer have intolerable debt, you should plug-in a *Contract-Oriented Cancel Criteria Algorithm* on your collection process templates. The CPM will call this algorithm if you've plugged it in.

Note: When all contracts are "removed" from a collection process, the CPM cancels all pending collection events and cancels the collection process.

Caution: Checking if individual contracts should be removed from a collection process is optional (meaning that you don't have to plug one in on the collection process template).

A New Payment Plan Can Cancel A Collection Process

Refer to *Collection Process Cancellation* for the details.

Note: Real time cancellation. Please be aware that the system will cancel a collection process real time when a promise to pay is created (if the promise to pay's scheduled payments are enough to pay-off the customer's outstanding debt).

A User May Cancel A Collection Process At Their Discretion

A user may cancel a collection process at their discretion.

Stopping A Contract May Cancel A Collection Process

The system will "remove" a contract from a collection process when it is stopped (i.e., when the contract's status becomes `Stopped`). When the last contract is "removed" from the collection process, the collection process will be cancelled.

The Big Picture Of Collection Events

This section describes the various types of collection events and their lifecycle.

How Are Collection Events Created?

Collection events may be created as follows:

- The Account Debt Monitor creates a collection process when an account violates collection criteria. The collection process has one or more collection event(s). The number and type of events is controlled by the collection process template associated with the collection process.
- Collection events are created when a user creates an ad hoc collection process. The number and type of events is controlled by the collection process template defined when the collection process is created.
- An ad hoc collection event may be created and linked to an existing collection process by a user at their discretion.

Note: Bottom line. Most collection events are created by the system when it creates a collection process for delinquent accounts. If you need to create an ad hoc collection event, you can either create a collection process whose template contains the desired event OR link the desired event to an existing collection process.

Fastpath: For more information about the creation of events by the Account Debt Monitor refer to *What Happens When A Collection Process Is Started?*. For more information about creating ad hoc collection processes, refer to *Collection Process Maintenance*. For more information about creating ad hoc events, refer to *Collection Process - Events*.

Types Of Collection Events

The following table describes the various types of collection events and what happens when they are completed:

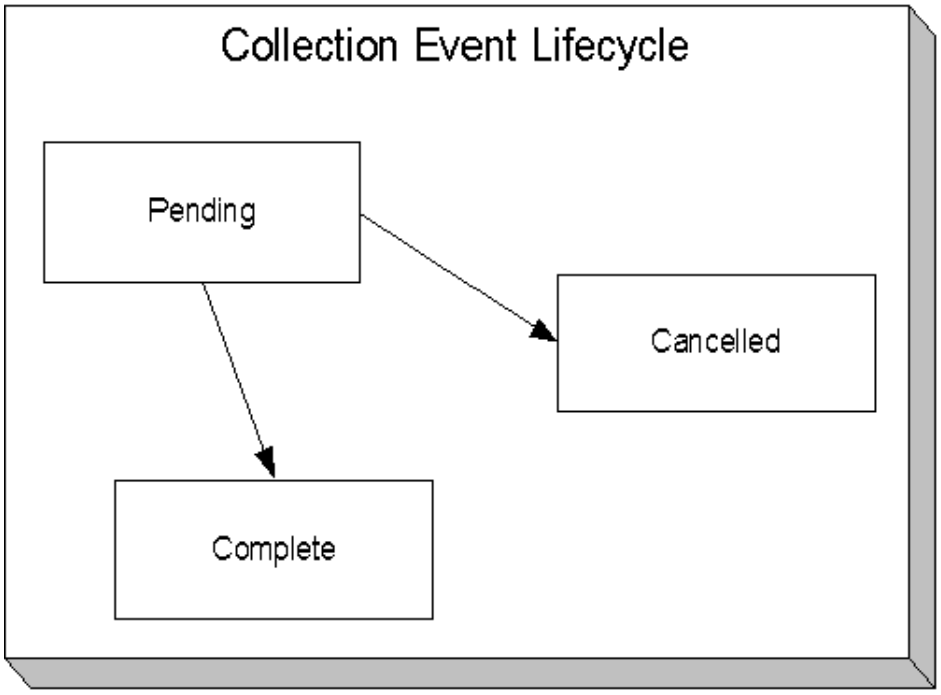
Type Of Collection Event	What Happens In The System
`Send Letter`	A customer contact is created for every financially responsible person linked to the account. The customer contact causes a letter to be produced. The type of letter is defined on the customer contact's contact type. The recipient of each letter is defined on *Account / Person* (those persons marked as **Receiving Notifications**).

Type Of Collection Event	What Happens In The System
`Create To Do Entry`	A To Do entry is created. Refer to *The Big Picture of To Do Entries* for more information about To Do entries.
`Affect Credit Rating/Cash-Only`	An account credit rating demerit record is created. The number of demerits is defined on the collection event type.
`Cancel Budget`	Every contract linked to the account that is on a budget is "removed" from the budget (i.e., the recurring charge amount for the contract is set to zero). In addition, a syncing adjustment is issued to cause each contract's current balance to be set equal to their payoff balance (the adjustment type is defined on the contract's contract type).
`Generic Algorithm`	The algorithm defined on the event's event type is executed.

Fastpath: Refer to *Setting Up Collection Event Types* for more information.

Collection Event Lifecycle

The following diagram shows the possible lifecycle of a collection event:



Collection events are initially created in the `pending` state.

When the system sees a `pending` event with a trigger date on or before the current date, the system executes the event's activity and `completes` the event.

Fastpath: For more information about a collection event's trigger date, see *Collection Event Trigger Date*.

A `pending` event will be `cancelled` automatically by the system when the account's debt no longer violates the collection criteria that sparked the event's collection process. A `pending` event may also be `cancelled` by a user at their discretion. Refer to *How Are Collection Processes Cancelled* for more information about how the system will cancel a collection process (and its events).

Collection Event Trigger Date

When a collection event is created by the system, its trigger date is set in accordance with your date arithmetic preferences. Refer to *Calendar vs. Work Days* for more information.

How Are Collection Events Completed?

A background process runs periodically (at least daily) that looks for collection events with a trigger date on or before the current date. For each triggered event, the system executes its activity and then completes it. Refer to *Collection Event Activator* for more information.

How Are Collection Events Canceled?

Users can cancel a collection event at their discretion. In addition, the system can cancel a collection event when it automatically cancels a collection process. Refer to *How Are Collection Processes Cancelled* for the details.

The Big Picture Of Write Off Processing

Before you financially write-off debt, most companies go to some effort to collect the past due funds. You control exactly what happens by setting up the various write-off control tables. The topics in this section provide background information that will help you understand how the information in these control tables is used.

How Is Debt Financially Written-Off?

Before debt can be written-off, a write-off contract must exist for the account. Why? Because when you write-off a normal contract's debt, you are actually transferring its debt to a write-off contract.

A write-off contract is just like other contracts in that:

- It holds debt.
- When a payment is received, the contract's debt is reduced.

Debt is transferred to a write-off contract (WO contract) from the customer's uncollectable contracts. The following points highlight important characteristics about the uncollectable contracts and the WO contract:

- The WO contract and the uncollectable contracts should be linked to the same account (note: this isn't a strict rule, it just makes sense because an account's written off funds should be linked to the account).
- Debt may be transferred to a WO contract from any type of contract regardless of debt class, i.e., a WO contract can contain debt that originated in any debt class.
- When you transfer debt from the uncollectable contracts to the WO contract, the debt is removed from the uncollectable contracts (and their status becomes `closed`- assuming their balance becomes zero).
- If you use the system's automated write-off processing, the system will create WO contracts for you. The system's automated write-off processing can write-off revenue in a different manner than is used to write-off liabilities. Refer to *The Ramifications of Write Offs in the General Ledger* for more information.
- WO contracts are immune from the account debt monitor (assuming their debt class is marked as not being subject to collection activities).
- WO contracts are not billed (assuming their contract type is marked as being not billable).

- WO contracts start their life with a non-zero payoff and current balances (i.e., they have debt when first started). This debt is transferred from the normal contract(s) whose uncollectable debt necessitated the creation of the WO contract.
- If the customer pays off the write-off debt, the WO contract remains active in case you ever need to write-off debt in the future. If you don't like the WO contract remaining active after it's paid off, you can indicate on the WO contract's contract Type that it is a "one time charge", this will cause the WO contract to be automatically closed when it's paid off.
- You can transfer additional uncollectable debt to the WO contract.

Note: Bankruptcy write-offs. If you have to write-off debt because a customer declares bankruptcy, everything stated above is true. The only thing you have to do is use a different contract type for bankruptcy write-offs as compared to "normal" write-offs. On the bankruptcy write-off contract type, simply leave the payment segment type blank - this way the system will never distribute a payment to the bankrupt debt (because bankrupt debt is legally uncollectable).

The Ramifications of Write Offs in the General Ledger

Caution: If you practice cash accounting, refer to *Cash Accounting and Write-Offs*.

When you write-off unpaid debt, you shouldn't book it all to a write-off expense account. Why? Because the debt that you're writing off typically contains both revenue and liabilities. At write-off time, you typically want to:

- Book the written off revenue to a write-off expense account, and
- Reduce the liabilities (you don't owe the liability if you don't get paid).

Consider the following example of a simple electric contract with two financial transactions:

Event	GL Accounting
Customer is billed	A/R 1000 Revenue <900> State Tax Payable - Taxing State - California <80> City Tax Payable - Taxing City - San Francisco <20>
Customer is levied a late payment charge	A/R 50 Late Payment Revenue <50>

After these two financial transactions are booked, the customer has debt of $1050. Of this $1,050; $950 is revenue and $100 is liability (money you owe the taxing authorities).

If the customer doesn't pay, you will eventually have to write-off this debt. Most organizations would issue the following types of financial transactions to do this:

Event	GL Accounting
Write-off the bill	Write-off Expense 900 State Tax Payable - Taxing State - California 80 City Tax Payable - Taxing City - San Francisco 20 A/R <1000>
Write-off the late payment charge	Write-off Expense 50 A/R <50>

Notice in the above transactions, the two separate revenue accounts are written off by booking to an expense account. However, the liability accounts are reversed. Why is revenue treated differently from liabilities at write-off time? There's a good reason for it (if you're an accountant), for the time being, just accept that this is how it works.

And finally, we need to worry about what happens if the customer eventually pays off his written off debt. If this happens, most organizations would pay off the write-off first, and, if there was still money left, they'd reimburse the taxing authorities. If we assume the customer pays off the entire written off debt, the following financial transactions would be issued:

Event	**GL Accounting**
Pay off the written off debt	Cash 900 Write-off Expense <900>
Reinstate the liabilities	Cash 100 State Tax Payable - Taxing State - California <80> City Tax Payable - Taxing City - San Francisco <20>

While the reinstatement of liabilities at payment time is possible in the system, the ramifications of doing such make this approach impracticable (the ramifications are a) if the check bounces, we would not be able to reduce the liabilities, and b) if there was a partial payment of the liabilities, the remaining unpaid amount could get written down). Therefore, when a write-off is paid the following financial transactions should be issued:

Event	**GL Accounting**
Pay off the written off debt	Cash 900 Write-off Expense <900>
Reinstate the liabilities	Cash 100 Reinstated liabilities <100>

Notice that rather than reinstating the individual liabilities, we simply reinstate all liabilities into a single account. This means your accountants will have to distribute this money to the appropriate liabilities manually.

So, how do we achieve the above in the system? This explanation is a little complicated, but it'll make sense if you keep the above financial transactions in mind:

- First of all, you'll need two different contract types - one to hold the written off revenue and another to hold the reduced liabilities.
    - On the contract type that holds written off revenue, indicate that it is not billable, indicate that it cannot have excess credits, and give it a high payment distribution priority. The distribution code on this contract type should reference your Write-off Expense account.
    - On the contract type that holds the reduced liabilities, indicate that it is not billable, indicate that it cannot have excess credits, and give it a high payment distribution priority. The distribution code on this contract type should reference a the "reinstated liabilities" GL account.

Next, you need to understand how the system's standard write-off logic works:

- The system accumulates the distribution codes from GL details associated with recent financial transactions linked to the contract being written-off.
- When the system has accumulated enough distribution codes (i.e., where the amount associated with the distribution code equals or exceeds the amount to write off), the debt will be transferred to a new or existing write-off contract(s). The number and type of contracts to which the bad debt is transferred is defined on the distribution codes. Refer to *Setting Up Distribution Codes* for how to define the type of write-off contract associated with a distribution code. In our example, we'd need the two contract types described above - one for the revenue accounts, the other for the liability accounts.

- At write-off time, for those distribution codes associated with revenue, the system will create a transfer adjustment from the normal contract to the write-off revenue contract. This will reduce (credit) the receivable on the normal contract and increase (debit) the expense account defined on the write-off revenue contract.
- However, if we do the above for the distribution codes associated with liabilities, we have a problem. The problem is a bit hard to explain unless you understand tax accounting, but it basically comes down to this - if we simply transfer the portion of the receivable balance associated with the liabilities to the write-off liability contract, we will always be debiting the distribution code defined on the contract type. This isn't correct because we really want to debit the liability account (and reference the characteristic type and value from the original credit) when we reduce the liability. So how do we do this? For those distribution codes associated with liabilities, you need to indicate that you want to override the distribution code on the "transfer to" side of the transfer adjustment with the distribution code / characteristic type / characteristic value that was originally booked. Refer to *Setting Up Distribution Codes* for how to indicate you want to override the distribution code at write-off time. If you do the above, then at write-off time the transfer adjustment will reduce (credit) the receivable on the normal contract and increase (debit) the original liability accounts from the original financial transactions.

If you followed the above, you'll see that we now have everything debited and credited appropriately. And, if a payment materializes for the written off debt, we will simply debit cash and credit the distribution code on the respective contract (either Write Off Expense or Reinstated Liabilities).

Note: Batch and real-time write-offs may use the above processing. The above logic is executed real time when a user writes off debt using the *write-off transaction* (assuming the base package *write off algorithm* is plugged into the account's *customer class*). The above logic is executed in batch when a write-off event that references a `Write Off Using Distribution Codes` *event type* is executed. Write-off events are described in detail below.

Automated versus Manual Write Offs

The system will automatically create write-off contracts and transfer uncollectable debt to them during the automated write-off processing described below.

If necessary, you can write-off debt outside of the automated write-off process using either of the following methods:

- You can transfer bad debt from any contract to a write-off contract using a transfer adjustment.
- You can use the *write-off transaction* to write-off debt real-time. When this transaction is used, the system executes the logic embedded in the `Write Off Method` algorithm that's plugged in on the account's *customer class*.

How Does The Write-Off Monitor Work?

This section describes how the *Write Off Monitor* uses your write-off criteria and write-off process templates to collect overdue debt.

Different Write-Off Criteria For Different Customers And Different Debt

Consider the following:

- You probably have different write-off criteria for different customer segments. For example, customers with large bills probably have strict criteria, whereas you're probably more lenient with small customers (or vice versa). You differentiate your customers in respect of the collection process via a **collection class code on the customers' accounts**. An account's initial collection class is defaulted from its customer class. You may override an account's collection class at will.
- You probably have different write-off criteria for different classes of debt. You differentiate your debt in respect of the collection process via a ***write-off* debt class on the customers' contracts** (note the *write-off* debt class is actually defined on the contract type and every contract has a contract type).

Note: Write Off Debt Class vs. Regular Debt Class. It's important to be aware that a contract type references both a regular debt class and a write-off debt class. The regular debt class controls the collection criteria applied against an account's contracts. The regular debt class is also used to segregate an account's outstanding balance on several queries in the system. The write-off debt class controls the write-off criteria applied against an account's stopped contracts. The reason the system supports two different debt classes is because you may categorize your contracts differently when you try to collect overdue debt versus when you write-off debt.

Given the above, you should understand that different write-off criteria will exist for every combination of collection class and write-off debt class. If you're confused, then consider the following matrix:

Account's Collection Class Contract's Write-Off Debt Class	Commercial Customer	Residential Customer
Regulated	N/A - there is no regulated, commercial customer debt.	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. Attempt to transfer debt to another active contract linked to the account. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment. If the debt is <= $-1, create an A/P adjustment to refund the credit to the customer. If debt remains, create the default write-off process for regulated debt.

Unregulated	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. Attempt to transfer debt to another active contract linked to the account. If the debt / credit is < $10 and > $-10, write down the debt using a write-down adjustment. If the debt / credit is <= $-10, create an A/P adjustment to refund the credit to the customer. If debt still remains: Highest priority: If customer has a non-cash deposit, create the non-cash deposit write-off process. Otherwise, create the default write-off process for unregulated commercial debt.	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. Attempt to transfer debt to another active contract linked to the account. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment. If the debt is <= $-1, create an A/P adjustment to refund the credit to the customer. If debt remains, create the default write-off process for unregulated residential debt.

Notice that each cell in the matrix has the same pattern:

- The system first attempts to reduce the contract's current and payoff balances to zero using the following methods (assuming you have set up the write-off control appropriately):
  - Sync the current balance with the payoff balance. If the contract's payoff balance is zero, this will cause the current balance to become zero and therefore close the contract.
  - If there's a debit balance, transfer the debt to any `pending start` or `active` contract in the same write-off debt class.
  - If there's a credit balance, transfer the debt to any `non-closed`/ `non-cancelled` contract in the same write-off debt class
  - If the remaining debit / credit balance is within a user-defined tolerance (this is defined on the respective algorithm on the write-off control), create an adjustment to write-down the small balance.
  - If a credit balance remains, create an A/P adjustment to refund the balance with a check (the adjustment type is defined on the respective algorithm on the write-off control).
- All of the above points will cause the contract to close. If debt remains, the system starts some type of write-off process. The type of process is dependent on the respective criteria. What differentiates one write-off criteria from another is its priority. The higher priority criteria will be compared first. If the customer / debt meets the criteria, the write-off process is initiated; no further comparisons are performed.

Fastpath: For more information about maintaining this matrix, refer to *Setting Up Write-off Control* .

When Is Debt Monitored For Write Off Purposes?

The write-off monitor only reviews a contract when the following conditions are true:

- The contract is `stopped` and `reactivated`.

- If the contract is a "billable charge" contract (as identified on its contract type), all of its billable charges must appear on a bill segment AND the bill segment's bill's due date plus grace period must be on or before the business date.
- If the contract is not a "billable charge" contract AND it is billable (as identified on its contract type), the contract must have a closing bill segment (i.e., it must be final billed) and the bill segment's bill's due date plus grace period must be on or before the business date.
- If the contract is a sub contract, its master contract must abide by the above conditions.
- If the contract is not billable, it is possible that adjustments, which affect the contract's debt, exist. The write-off monitor will only review a non-billable contract if all FTs for this contract that have been marked to include on a bill have been swept onto a bill and the bill for any of these FTs has a bill due date plus grace period on or before the business date.

Note: Postponing write-off processing. You can prevent the write-off process from processing an eligible contract by populating the account's C&C Postpone Date with a future date.

Attempt To Close The Contract Before Creating A Write Off Process

Before the write-off monitor creates a write-off process for a `stopped` and `reactivated` contract, it attempts to reduce the contract's debt to zero using all of the following methods:

- If the account has active contracts, it will transfer the finaled debt to a pending start or active contract.
- If the debt or credit amount on the contract is small, the system will generate an adjustment to 'write it down' (or up in the case of a small credit).
- If the contract has a large credit amount, the system will generate an A/P adjustment (resulting in a check being sent to the customer).

Note: Plug-in algorithms do the work. Algorithms that are plugged-in on the *write-off control* responsible for managing the contract's debt actually perform the above effort. You can customize these algorithms to behave exactly how your collections staff desires.

If the algorithms responsible for the above effort are successful in reducing the contract's debt to zero, then the contract `closes` and will not be subject to write-off processing. If the above algorithms don't result in the contract's debt being reduced to zero, a write-off process will be started (as describe below).

What Happens When A Write-Off Process Is Started?

When you define write-off criteria, you must define the write-off process template to use if the criteria are violated. The system uses this template to create the account-specific write-off process.

Every `stopped` or `reactivated` contract that is part of the offending write-off debt class will be linked to the write-off process.

Also linked to the write-off process will be one or more write-off events. These events are meant to prod the customer. You define the types of events and when they are generated when you set up your write-off process templates.

It's important to note that all of the write-off events will be created when the write-off process is created. Each of these write-off events contains a trigger date. The trigger date of the first event(s) will typically be the current date. The trigger date of the other events will be in the future. Refer to *Calendar vs Work Days* for a description of how the trigger date is calculated.

A separate process, Activate Write-off events, is responsible for activating write-off events whose date is on or before the current date. Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do to an operator, refer debt to a collection agency, etc.)

Note: Multiple write-off processes may be kicked off. It's important to be aware that if an account's contracts reference multiple write off debt classes, a write-off process will be started for each offending write off debt class that has `stopped` or `reactivated` contracts.

Fastpath: For more information about write-off process templates, see *Setting Up Write Off Process Templates* . For more information about write-off events, see *The Big Picture Of Write-off Events* . For more information about how a write-off process is cancelled, see *How Does A Write-Off Process Get Cancelled?* .

How Does A Write-Off Process Get Cancelled?

The system "removes" a contract from a write-off process when its status becomes `closed` (i.e., when its balance is zero). When all contracts are removed, the system cancels all pending write-off events and deactivates the write-off process. When the write-off process is deactivated, all collection agency referrals associated with the write-off process are cancelled.

Note: Removing closed contracts from a write-off process. Contracts aren't actually removed from the process. Rather, they are inactivated so a proper audit exists.

How Do Collection Agency Referrals Work?

The following points describe how collection agency referrals work.

- A write-off process has one or more events. One type of event causes overdue debt to be referred to a collection agency.
- When a referral write-off event is activated, the system marks the event for processing by the event's Collection Agency Referral Algorithm (refer to *Setting Up Write Off Event Types* for more information).
- The next time the Collection Agency Referral process executes (the frequency is dependent on your background process schedule), it will refer the process' debt to a collection agency. The specific agency to which the debt is referred is controlled by the event type's Collection Agency Referral Algorithm. The sample algorithm supplied with the system simply refers debt to the collection agency with the least amount of referred debt. If you prefer different logic, you must write your own algorithm.
- Regardless of the manner in which a collection agency is selected for an account's debt, the referral involves the creation of a collection agency referral history record. Refer to *Collection Referral* for more information.
- A collection agency referral history record is linked to an account. It contains the amount of debt referred to the collection agency. It is the creation of this record that, in turn, triggers the interface of information to the collection agency. The method used to interface the information to the agency is defined on the collection agency's record. Refer to *Setting Up Collection Agencies* for more information.
- If the collection agency is successful in obtaining the funds, simply add a payment. If the payment causes the contract's balance to become zero, the system will automatically `close` the contract. When the system closes a contract, it is "removed" from the write-off process. When a write-off process no longer contains active contracts, the system cancels the write-off process. When a write off process is cancelled, all collection agency referrals are automatically cancelled.
- Collection agency referrals get cancelled by the creation of a new collection agency referral history record (with a type of `cancel`). This record will be interfaced to the agency in the same manner used to interface a new referral (see above).
- If the collection agency is not successful in obtaining your funds after a given amount of time, you probably want to cancel the referral and write-off the debt. The cancellation of the referral will happen automatically if you design your write-off process to generate a collection agency cancellation X days after the referral. Refer to *Setting Up Write Off Process Templates* for how to do this. You can cancel a referral manually by simply creating a new collection agency referral history record (with a type of `cancel`).

Fastpath: When you enable the Control Central alert algorithm, *C1-COLL-REF*, an alert displays when an account has an active collection agency referral. This algorithm is plugged-in on the *installation record*.

The Big Picture Of Write-off Events

This section describes the various types of write-off events and their lifecycle.

How Are Write-off Events Created?

Write-off events may be created as follows:

- The Write-Off Monitor creates a write-off process when an account has unpaid, final billed contracts. The write-off process has one or more write-off event(s). Refer to *How Does The Write-Off Monitor Work?* for more information about how the system creates write-off processes and their events.
- Write-off events are created when an operator creates an ad hoc write-off process. The number and type of events is controlled by the write-off process template defined when the write-off process is created.
- An ad hoc write-off event may be created and linked to an existing write-off process by an operator at their discretion.

Note: Bottom line. Most write-off events are created by the system when it creates a write-off process for unpaid, finaled contracts. If you need to create an ad hoc write-off event, you can either create a write-off process using a template that contains the desired event OR link the desired event to an existing write-off process.

Fastpath: For more information about creating ad hoc write-off processes and events, refer to *How To Perform Common Write-off Maintenance Functions* .

Types Of Write-off Events

The following table describes the various types of write-off events and what happens when they are completed:

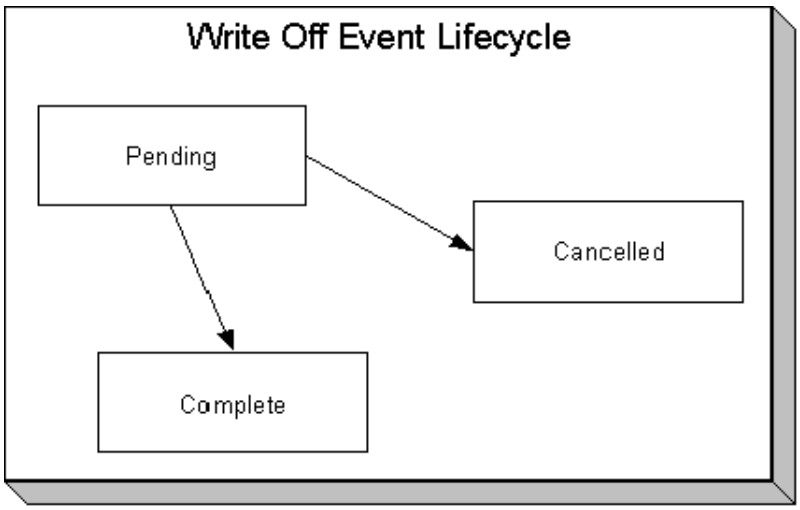
Type Of Write-off Event	What Happens In The System
`Affect Credit Rating/Cash-Only`	An account credit rating demerit record is created. The number of demerits is defined on the write-off event type.
`Cancel Agency Referral`	All collection agency referrals associated with the write-off process will be cancelled.
`Refer to Agency`	The debt associated with the contract's linked to the write-off process will be referred to a collection agency.
`Send Letter`	A customer contact is created for every financially responsible person linked to the account. The customer contact causes a letter to be produced. The type of letter is defined on the customer contact type control table. The recipient of each letter is defined on *Account / Person* (those persons marked as **Receiving Notifications**).

Type Of Write-off Event	What Happens In The System
`Send To Do`	A To Do entry is created. Refer to *The Big Picture of To Do Entries* for more information about To Do entries.
`Write Off using Distrib Code`	This type of event is used to write-off bad debt in accordance with the distribution codes associated with the financial transactions that caused the debt in the first place. You'd use this method for example if you want to write-off revenue differently than you write-off liabilities. The system accumulates the distribution codes from GL details associated with recent financial transactions linked to each write-off contract. When the system has accumulated enough distribution codes (i.e., where the amount associated with the distribution code equals or exceeds the amount to write off), the debt will be transferred to a new or existing write-off contract. The type of contracts to which the debt is transferred is defined on the distribution codes.
`Write Off using Contract Type`	The contracts linked to the process will be written-off by transferring their debt to a new or existing write-off contract. Note: the contract type of the write-off contract is defined on the algorithm defined on events of this type.
`Generic Algorithm`	The system calls the algorithm defined on the write-off event type. This type of event is used when what you need to do isn't handled by one of the above event types.

Fastpath: Refer to *Setting Up Write Off Event Types* for more information.

Write-off Event Lifecycle

The following diagram shows the possible lifecycle of a write-off event:



Write-off events are initially created in the `pending` state.

When the system sees a `pending` event with a trigger date on or before the current date, the system executes the event's activity and `completes` the event.

Fastpath: For more information about a write-off event's trigger date, see *Write-off Event Trigger Date* .

A `pending` event will be `cancelled` automatically by the system when the contracts linked to the process are all `closed` (i.e., they no longer have debt - either because it was paid or transferred to a write-off contract). A `pending` event may be `cancelled` by an operator at their discretion.

Write-off Event Trigger Date

When a write-off event is created by the system, its trigger date is set in accordance with your date arithmetic preferences. Refer to *Calendar vs. Work Days* for more information.

How Are Write-off Events Completed?

A background process runs periodically (at least daily) that looks for write-off events with a trigger date on or before the current date. For each triggered event, the system executes its activity and then completes it. Refer to *Write Off Event Activator* for more information.

The Last Write-off Event Should Write Off All Debt Associated With All Contracts

The last write-off event will typically transfer all debt from the contract's linked to the process to a write off contract (linked to the account). This will only happen if you set up the write-off process template accordingly (i.e., the last event in the write-off process template is the kind that writes off debt for every contract linked to the write-off process).

How Are Write-off Events Canceled?

The system removes a contract from a write-off process when its status becomes `closed` (i.e., when its balance is zero). Be aware that any type of financial event could cause an contract's balance to fall to zero (e.g., the creation of an adjustment, the application or cancellation of a payment, the cancellation of a bill,...). When all contracts are removed, the system cancels all pending write-off events and deactivates the write-off process.

Note: Real time cancellation. Unlike collection processes, the system cancels write-off processes real time when the contract becomes `closed` (i.e., there is no background process that monitors write-off processes).

Besides the automated cancellation process, an operator may cancel a write-off event at will.

Note: Removing `closed` contracts from a write-off process. contracts aren't actually removed from the process. Rather, they are inactivated so a proper audit exists.

Calendar vs. Work Days

When you set up your collection and write-off process templates, you supply information that controls how the system determines the trigger date of each event in the related process. You can do this as follows:

- When you set up your collection and write-off process templates, you must define the number of days after the start of the process when each event should be triggered. For example, the 2 nd event (send cutoff warning) may need to be triggered 7 days after the start of the collection process.

The system uses this information in conjunction with the account's division's work calendar when it allocates a trigger date to the various collection and write-off events in your processes. The system offers you the following choices in respect of how it calculates an event's trigger date:

- You can indicate that the trigger date should be set to the next possible workday. For example, if you indicate that the 2 nd event is triggered 7 days after the 1 st event, the system will add 7 days to the 1 st event's completion date. It then checks if this is a workday (and not a holiday), if so, this is the trigger date of the event; if not, it assigns the trigger date to the next workday.
- You can indicate that the trigger date should be calculated by counting workdays. For example, if you indicate that the 2 nd event is triggered 7 days after the 1 st event, the system will count 7 workdays (using the account's division's work calendar), and set the trigger date accordingly.

You must define which of the above methods is used in the following processes:

- Account Debt Monitor (ADM and ADM2). Refer to *The Account Debt Monitor* for more information.
- Collection Event Trigger (CET). Refer to *The Collection Event Activator* for more information.
- Write-off Monitor (WPM). Refer to *The Write Off Monitor* for more information.

The Big Picture Of Payment Arrangements and Promise To Pay

The topics in this section describe two different mechanisms that allow a customer to payoff overdue debt in installments.

The Big Picture Of Pay Arrangements

A payment arrangement is an agreement with a customer to payoff severely overdue debt in ***billed*** installments. Bills sent to customers with payment arrangements contain charges for both their current services and their payment arrangement installment amount.

Note: Nomenclature. Some people refer to payment arrangements as "current bill plus" agreements because the customer's bills contain charges for both their current debt plus their installment amount. After the customer has paid off their overdue debt, the customer's bill only contains charges for their current debt.

The topics in this section describe how to set up a payment arrangement and how the system monitors the ongoing arrangements.

Creating Payment Arrangements

When you create a payment arrangement, you are actually creating a contract. This contract is just like other contracts in that:

- It holds debt.
- It is periodically billed.
- When a payment is received, the contract's debt is reduced.
- If the contract becomes delinquent, a collection process is initiated to collect the overdue debt.

Debt is transferred to a payment arrangement contract (PA contract) from the customer's delinquent contracts at the inception of the payment arrangement.

When you transfer delinquent debt from the delinquent contracts to the PA contract, the debt is removed from the delinquent contracts. If you transfer all debt from the delinquent contracts, the customer will no longer be in arrears in a given debt class (and if the customer is no longer in arrears, active collection process will be cancelled).

Note: Use the Payment Arrangement Transaction. You could do the above functions by adding a new contract and creating transfer adjustments. However, this is tedious. Rather, use the *Payment Arrangement* transaction. This transaction creates the PA contract, transfers debt to it, and sets up the installment amount. This transaction is also used if you need to break or cancel the payment arrangement.

Installment, Payoff and Current Amounts

Caution: If you do not understand the difference between payoff balance and current balance, refer to *Current Amount versus Payoff Amount* .

When you set up a payment arrangement contract (PA contract), you transfer delinquent debt to the PA contract using transfer adjustments. After moneys are transferred, the system sets the PA contract's *current balance* to zero. At this point, neither the original contracts nor the PA contract have delinquent debt. If the customer neglects to pay their payment arrangement, the PA contract will fall into arrears and a collection process will ensue. If the customer neglects to pay their previously delinquent contract's, they will again fall into arrears and a collection process will ensue.

PA contract's start their life with a non-zero *payoff balance* (i.e., they have debt when first started). This debt is transferred from the normal contract(s) whose outstanding debt necessitated the creation of the PA contract.

The installment amount that the customer is billed is determined by the number of installments used to payoff the debt. The installment amount is saved on the PA contract's recurring charge amount. If the customer again falls into arrears on their normal contracts, you can transfer additional delinquent debt to the PA contract. You can also change the installment amount as needed.

A PA contract's payoff balance typically differs from its current balance. The payoff balance is the amount of debt remaining to be paid off under the terms of the payment arrangement. The current balance is the installment amount that has been billed but not paid. For example, a customer who is paying off $500 with 10 installments of $50 would have an initial payoff balance of $500 and a current balance of $0. After the first bill, the PA contract would still have a payoff balance of $500, but its current balance would be $50. When the customer pays, the PA contract's payoff balance would fall to $450 and its current balance would return to $0.

The following table contains a financial example of a customer who sets up a payment arrangement to payoff $1,000 of debt in $10 installments.

Event	**Normal Contract's GL Accounting**	**PA Contract's GL Accounting**	**Normal Contract's Current Balance**	**Normal Contract's Payoff Balance**	**PA Contract's Current Balance**	**PA Contract's Payoff Balance**
Prior to creation of payment arrangement	N/A	N/A	1000	1000	N/A	N/A
Transfer debt from normal contract(s) to PA contract	Xfer 1000 A/R <1000>	PA A/R 1000 Xfer <1000>	0	0	1000	1000

Event	Normal Contract's GL Accounting	PA Contract's GL Accounting	Normal Contract's Current Balance	Normal Contract's Payoff Balance	PA Contract's Current Balance	PA Contract's Payoff Balance
Set current balance to zero on PA contract	N/A	N/A	0	0	0	1000
Customer is billed ($50 for new debt and $10 of payment arrangement debt)	A/R 50 Revenue <50>	N/A	50	50	10	1000
Customer pays $60	Cash 50 A/R <50>	Cash 10 PA A/R <10>	0	0	0	990

When the customer pays off the payment arrangement debt, the system automatically closes the PA contract after it final bills (assuming the PA contract's contract type references a bill segment type that has a bill segment creation algorithm of `Recurring Charge With Auto Stop`).

Monitoring Payment Arrangements

The PA contract should belong to its own debt class (let's call it Payment Arrangement Debt) so that you can have stricter collection criteria for payment arrangement debt (as compared to normal contract's). Because there will be a new debt class, there will be a unique collection class control (CCC) for payment arrangements. This CCC will have debt criteria associated with payment arrangement debt.

Note: The PA contract must final bill before it closes. It's important to note that the PA contract will only close after the PA contract is final billed. This is OK as it won't have any money left on it.

When the ADM next runs, it will analyze the account's reinstated debt. We recommend creating a new override collection criteria for the normal debt class that will return a value of true if the account has a closed payment arrangement that has been broken in the last X days (where X is a parameter of the override collection criteria's algorithm). If this algorithm returns a true, kick off a unique collection process template (that has nasty events). A sample algorithm of this type is supplied in the base package - `COLL COND PA`.

To complete this discussion, we have to worry about the situation when the final bill of a payment arrangement goes unpaid. In this situation, the payment arrangement is stopped and will therefore not be processed by the ADM. In this case, the write off monitor will process the PA contract after its final bill's due date and a write-off process will start. This write off process will have a single event that calls the Break Payment Arrangement algorithm (described above). After the FT's are issued in this event, the contract will close (because it's been final billed and it's balance will go to zero).

The Big Picture Of Promise To Pay

A promise to pay (PTP) is an agreement with a customer to make payments on specific dates. Promise to pay differs from payment arrangements in that a promise to pay contains user-defined scheduled payment dates, which are independent from the customer's billing dates. In other words, payment arrangements appear on the customer's bills, promise to pay scheduled payments do not.

If a customer is in arrears and you want to receive payments on specific dates (as opposed to with the customer's regular bills), you would set up a promise to pay and define the dates on which you expect the payments.

The topics in this section describe how a promise to pay works.

A Promise To Pay Has One Or More Scheduled Payments

When you create a promise to pay for an account, you must define the number of scheduled payments and their respective amounts. There is no limit to the number of scheduled payments that may be set up under a promise to pay.

Automatic Payments Can Be Created On The Scheduled Payment Dates

The system will create automatic payments on a promise to pay's scheduled payment dates if:

- The account is set up for automatic payment (as described under *How To Set Up A Customer To Pay Automatically*), and
- The pay method defined on the promise to pay indicates automatic payment is being used

The background process called PPAPAY is responsible for creating these automatic payments. It does this by calling the automatic payment creation algorithm plugged in on the installation record.

Please note, if the **Autopay Creation Option** on the *installation record* is set to `Create On Extract Date`, the automatic payment is NOT distributed and frozen when the automatic payment is initially created. Rather, a separate background process (*APAYDSFR*) distributes and freezes the automatic payment on the automatic payment GL distribution date (refer to *Automatic Payment Dates* for more information on how this date is calculated). Refer to *Automatic Payments* for more information.

A Promise To Pay Insulates Overdue Debt From The Account Debt Monitor (ADM)

A promise to pay's scheduled payments are used by the account debt monitor as "pseudo payments" that relieve the account's debt before it is subjected to the collection criteria (refer to *How Does The Account Debt Monitor Work* for more information about collection criteria).

It's important to understand that a promise to pay only insulates the account's debt that belongs to the promise to pay's debt class. Therefore, if a customer has debt that belongs to two debt classes (e.g., normal debt and 3 rd party pass through debt), you would need to set up a separate promise to pay for each debt class (assuming both types of debt are covered by a promise to pay). Refer to *Different Collection Criteria For Different Customers and Different Debt* for more information about debt classes.

A Promise To Pay Must Reference A Promise To Pay Type

When you create a promise to pay, you must define its promise to pay type. The promise to pay type controls the following functions:

- The debt class whose debt is insulated by the promise to pay.
- The type of algorithm (if any) that is executed when the promise to pay is broken. You might use such an algorithm to affect the customer's credit rating when the promise to pay is broken.

A Promise To Pay May Reference A Third Party Payor

In addition to referencing the account whose debt is insulated by the promise to pay, the promise to pay must also reference the account that is responsible for making the payments. We refer to this second account as the promise to pay's "payor".

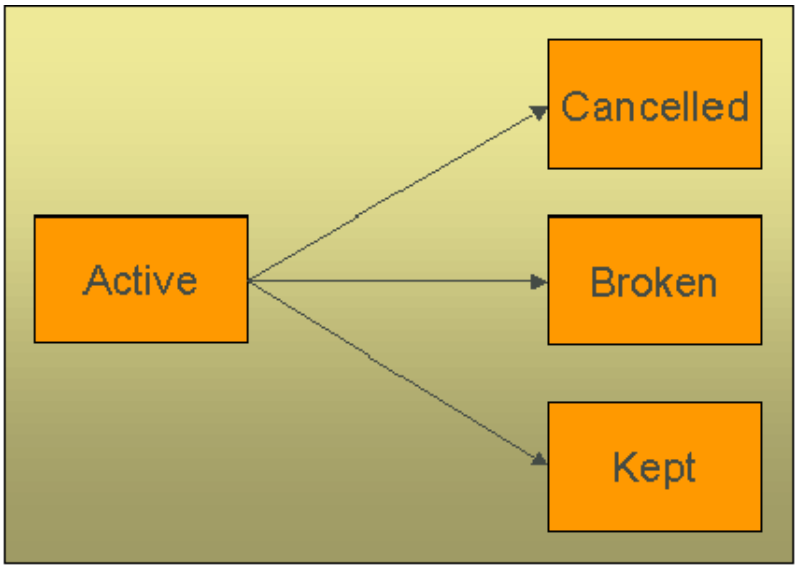
While the payor's account is typically the same as the account whose debt is insulated by the promise to pay, you can indicate a third-party payor (e.g., a social service agency) is responsible for making the promise to pay's scheduled payments.

If your organization allows third-party payors, you can define each on the third-party payor control table. This control table exists to simplify the data-entry effort when you create a promise to pay (as it defines the account associated with the third-party payor).

Note: If a promise to pay does not reference a third-party payor, any *non-third-party payors* (i.e., any account that is not defined in the third-party payor control table) can make payments on behalf of the customer. If a promise to pay references a third-party payor, only payments made by the third party on behalf of the customer are counted towards the fulfillment of the promise to pay.

The Lifecycle Of A Promise To Pay

The following diagram shows the possible lifecycle of a promise to pay:



The following points explain this lifecycle:

- Promise to pay is initially created in the `active` state. `Active` promise to pay is monitored for compliance by *The Promise To Pay Monitor*.
- A promise to pay may be `cancelled` as follows:
  - A user can cancel a promise to pay at will.
  - When a contract is stopped AND there are no other active contracts in the same debt class, all `active` promise to pay associated with the account and debt class will be `cancelled`.
  - The activation of a collection event that calls the "cancel promise to pay" algorithm will cancel all `active` promise to pay associated with the collection process's debt class. You may want to use such a collection event if your organization cancels `active` promise to pay when new debt causes a collection process to kick-off. Note, the base package algorithm that performs this function will not cancel the promise to pay if it's associated with a 3 rd party payor.
- *The Promise To Pay Monitor* causes `active` promise to pay to become `broken` if sufficient payments have not been made to satisfy the promise to pay's scheduled payments.
- *The Promise To Pay Monitor* causes `active` promise to pay to become `kept` when it detects that sufficient payments have been made to satisfy the promise to pay's scheduled payments.

Highlighting The Existence Of Broken / Kept / Active and Denied Promise To Pay

You can define on the installation record plug-in algorithms that format alert messages. (Refer to *Installation Options - Algorithms* for additional information.) We recommend that you take advantage of the following algorithms to highlight promise to pay:

- Highlight promise to pay in a given status. This algorithm is used to highlight promise to pay in a given state (broken, kept, cancelled) that were started within the last X days.
- Highlight customer contacts of a given type. This algorithm would be used to highlight customer contacts of a given type that were created within the last X days. This would be useful if you create a specific type of customer contact when you deny a promise to pay.

In addition, you can define account-specific alerts to highlight customers that should never be allowed to have a promise to pay (for whatever reason).

A Promise To Pay Must Reference A Payment Method

When you create a promise to pay, you must define how the customer will make the payments be referencing a payment method. Examples of payment methods include: `In Person`, `Wire Transfer`, `By Post`, `Express Mail`, etc.

The payment method is more than just documentation as it defines the number of grace days the customer has to make the promise to pay's scheduled payments. For example, if you set up the payment method control table to indicate that payments made `By Post` have 3 grace days, then the customer has up to 3 days after each scheduled payment date to make the payment. If payment is not received by the scheduled payment date plus the grace days, the promise to pay will be marked as `broken` (and the ADM will be triggered).

The Promise To Pay Monitor

Fastpath: Please understand the concepts described in *The Lifecycle Of A Promise To Pay* and *The Tendering Account May Differ From The Account Whose Debt Is Relieved* before reading this section.

The Promise To Pay Monitor background process (referred to as `PPM`) is responsible for monitoring `active` payment plans. This process can cause a promise to pay (PTP) to become `kept` or `broken` (or being left as `active`).

Note: When is a promise to pay marked as broken / kept? It's important to understand that only the `PPM` can cause a promise to pay to become `kept` or `broken`. This means that if a customer makes a payment that satisfies a promise to pay, the promise to pay will only be marked as `kept` when the promise to pay monitor next runs. Analogously, if a payment is `cancelled`, nothing will happen to an `active` promise to pay until the `PPM` next runs. When the `PPM` next runs, it will see that the scheduled payment was not kept and it will break the promise to pay and schedule the ADM to be executed. When the ADM next executes, it will create a collection process (because the customer's debt will no longer be insulated by the promise to pay's scheduled payments).

Note: NSF Cancellations After A Promise To Pay Is Kept. If a payment is `cancelled` due to non-sufficient funds (NSF) after a promise to pay is marked as `kept`, the promise to pay will remain `kept`. But keep in mind that the promise to pay's account is scheduled for review by the ADM when a payment is `cancelled` due to NSF. When the ADM reviews the account's debt, it will no longer have an `active` promise to pay to insulate it and the account's debt will likely trigger a new collection process. Refer to *How Promise To Pay Affect The ADM* for more information.

The following points describe, at a high level, how the `PPM` monitors a promise to pay (PTP) for compliance.

- The system selects all `frozen`, non-cancelled payment segments associated with the PP's account and debt class where:
  - The payment date is after the start date of the promise to pay, and
  - The payment's pay event has at least one tender that references the promise to pay's payor.
- The system logically reduces / removes past and current scheduled payments (starting with the earliest scheduled payment) until the total amount of payment segments is exhausted (or there are no more historical / current scheduled payments).

Note: Paying promise to pay in advance. Scheduled payments with a future date are not logically removed / reduced. This means that if a customer makes advance payments on a promise to pay, it will not be marked as `kept` until all scheduled payment dates are in the past.

- If all scheduled payments have been logically removed, the promise to pay is marked as `kept`.
- If there exist scheduled payments where the pay date + grace days (grace days are defined on the promise to pay's payment method) is before the current date (i.e., a payment doesn't exist for a scheduled payment):
  - The promise to pay is marked as `broken`.

- The PP's break algorithm (if any) is called (note, for European / Australian promise to pay, there are scenarios where the break algorithm can cause the promise to pay to become unbroken - when there aren't at least two missed, historical scheduled payments).
- An ADM trigger is stored for the PP's account. This will cause the account to be reviewed by the ADM the next time it runs. And because the promise to pay is broken, its scheduled payments will no longer insulate the account's arrearage.

Note: Important! It's important that you schedule the PPM to run before the ADM so that it can break unpaid payment plans prior to the ADM subjecting the account's debt to collection criteria. Refer to *How Promise To Pay Affect The ADM* for more information.

How Promise To Pay Affect The ADM

As described under *A Promise To Pay Insulates Overdue Debt*, a promise to pay's scheduled payments insulate an account's debt from the ADM. This section describes how this is accomplished.

Caution: You should understand the concepts in *How Does The Account Debt Monitor Work* and *The Tendering Account May Differ From The Account Whose Debt Is Relieved* before reading the following.

Note: The ADM will be triggered when a promise to pay is broken. Refer to *The Promise To Pay Monitor* for an explanation of how the ADM is triggered when a promise to pay is broken.

Before the ADM (and ADM2) subjects an account's debt to the collection criteria, it calls the debt's debt class's Override Arrears Algorithm (this is an optional plug-in spot on *Debt Class*). This algorithm is passed the debt class's aged debt and manipulates it as follows:

- First, a list of all past, present and future scheduled payments associated with the account and debt class's `active` promise to pay is constructed.
  - If multiple payors are encountered (because the customer has multiple promise to pay and these have different payors), a separate list of scheduled payments is maintained for each payor.
- Next, for each payor, retrieve the total amount of `frozen`, non-cancelled payment segments made on behalf of the promise to pay's account and debt class.
  - Select all `frozen,` non-cancelled payment segments associated with the promise to pay's account and debt class whose pay date is >= promise to pay's start date and the pay segment's event has at least one tendering account associated with the promise to pay's payor.
- Next, logically reduce / remove past and current scheduled payments (starting with the earliest scheduled payment) until the payor's payment amount is exhausted (or there are no more historical / current scheduled payments). Future scheduled payments cannot be remove / reduced.
- Finally, reduce the passed in aged debt with any unpaid scheduled payments.

Note: This logic is not "hard coded". Rather, the mechanism used to use a promise to pay's scheduled payments to reduce debt is defined in an algorithm defined on the promise to pay's debt class. The contents in this section describe how a base package algorithm works. Because it's an algorithm, you can introduce whatever logic you please.

The following is an example of how promise to pay affect aged debt.

Date	**Event**	**Contract's Arrears**	**'s Balances**	**Scheduled Payments**
Prior to creation of the PP 1/18/2000		$1,000 - 90 days old $1,600 - 60 days old $1,900 - 30 days old	Current: $4,500 Payoff: $4,500	
1/18/2000	Promise to pay created. The $4,500 in future scheduled payments offsets the existing $4,500 of aged debt.	$1,000 - 90 days old $1,600 - 60 days old $1,900 - 30 days old De facto ADM debt: $0	Current: $4,500 Payoff: $4,500	1/20/2001 $1,500 2/01/2001 $1,500 2/07/2001 $1,500
1/20/2001	The customer pays $1,500. There exists $3,000 of future scheduled payments that offset the arrears	$1,100 - 62 days old $1,900 - 32 days old De facto ADM debt: $0	Current: $3,000 Payoff: $3,000	1/20/2001 $1,500 "Paid" 2/01/2001 $1,500 Future 2/07/2001 $1,500 Future
1/20/2001	ADM runs. The $3000 in future scheduled payments offsets the Current Balance of $3000, so CC events not created.	$1,100 - 62 days old $1,900 - 32 days old De facto ADM debt: $0	Current: $3,000 Payoff: $3,000	1/20/2001 $1,500 "Paid" 2/01/2001 $1,500 Future 2/07/2001 $1,500 Future
1/24/2001	A new bill is created for $400	$1,100 - 62 days old $1,900 - 32 days old $400 - 1 day old De facto ADM debt: $400 - 1 day old	Current: $3,400 Payoff: $3,400	1/20/2001 $1,500 "Paid" 2/01/2001 $1,500 Future 2/07/2001 $1,500 Future

Date	Event	Contract's Arrears	's Balances	Scheduled Payments
2/2/2001	Promise To Pay Monitor runs. PP marked as `Broken` because the 2/1/2001 scheduled payment has not been paid (assuming no grace period on the promise to pay's payment method)	$1,100 - 74 days old $1,900 - 44 days old $400 - 8 days old	Current: $3,400 Payoff: $3,400	1/20/2001 $1,500 "Paid" 2/01/2001 $1,500 "Late" 2/07/2001 $1,500 Future
2/2/2001	ADM runs. There are no active promise to pay and therefore there is nothing to insulate the customer's debt. Therefore the aged debt will be subjected to the collection criteria and an appropriate collection process will be created.	$1,100 - 74 days old $1,900 - 44 days old $400 - 8 days old De facto ADM debt is the same as above (i.e., rather old)	Current: $3,400 Payoff: $3,400	Promise to pay is broken and therefore its scheduled payments cannot be used.

Collection Process Cancellation

When a promise to pay (PTP) is created, the system determines if it can cancel `active` collection process associated with the promise to pay's account and debt class. It does this because a promise to pay's scheduled payments act as "pseudo payments" that relieve the account's debt (temporarily). The following points describe how this works:

- The system attempts to cancel collection processes by calling the *Collection Process Cancel Criteria Algorithm* defined on the debt class that is associated with the collection process. This algorithm is meant to cancel a collection process if the sum of ALL contracts in the debt class have debt less than a given threshold amount. Because of the existence of the promise to pay, the actual debt will be temporarily reduced by the amount of the promise to pay's scheduled payments before it is compared to the threshold amount (see *How Promise To Pay Affect The ADM* for more information about how debt is reduced). Note: this temporary reduction will only occur if you have plugged in the appropriate promise to pay debt reduction algorithm on the debt class.

If collection processes still exist for the account / debt class associated with the promise to pay, a warning is issued.

Interesting Promise To Pay Facts

The following points describe a variety of interesting facts about promise to pay (PTP):

- An account may have many active promise to pay. However, only 1 promise to pay may be active for a given account / debt class / payor at any point in time.
- The existence of a promise to pay has no impact on payment distribution.

- When a contract is stopped, if the contract's debt class has an active PP AND there are no other active contract's in the same debt class, the PP will be cancelled.
    - The cancel reason will be "cancelled by system" (as opposed to "cancelled by user")
- If necessary, different collection processes can be triggered if a broken PP is detected (via the override algorithms on CCC and write-off control - we do NOT provide such algorithms).

Setting Up Promise To Pay Control Tables

This section describes the control tables needed to set up promise to pay.

Setting Up Promise To Pay Types

Promise To Pay Types control what is done by a given promise to pay. Open **Admin, Promise To Pay Type** to define your promise to pay types.

Fastpath: For more information refer to *The Big Picture Of Promise To Pay* for more information.

Description of Page

To modify a promise to pay type, simply move to a field and change its value. To add a new promise to pay type, press + to insert a row, then fill in the information for each field. The following fields display:

Promise To Pay Type The name of the promise to pay type.

Description A meaningful description of the promise to pay type.

Broken Algorithm This algorithm is called when a promise to pay is broken. Refer to *The Promise To Pay Monitor* for more information about how promise to pay is broken.

If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to *Setting Up Algorithms*). On this algorithm, reference an Algorithm Type that breaks a promise to pay. Click *here* to see the algorithm types available for this plug-in spot.

Debt Class The debt class covered by promise to pay of this type. Refer to *Setting Up Debt Classes* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_PP_TYPE*.

Setting Up Payment Methods

Payment methods are used to describe how a customer intends to make their promise to pay's scheduled payments. Open **Admin, Pay Method** to define your payment methods.

Fastpath: For more information refer to *A Promise To Pay Must Reference A Payment Method* for more information.

Description of Page

To modify a pay method, simply move to a field and change its value. To add a new pay method, press + to insert a row, then fill in the information for each field. The following fields display:

Pay Method The name of the payment method.

Description A meaningful description of the payment method.

Grace Days The number of days added to the scheduled payment date. The ADM will consider the promise to pay to be broken if payment is not made by the scheduled date plus the grace days.

Auto Pay If the pay method is marked as being for **Auto Pay**, the PPAPAY background process will automatically create an automatic payment on the promise to pay's scheduled payment dates IF the account has been set up for automatic payment.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_PAY_METH*.

Setting Up Third Party Payors

Promise to pay supports optional third-party payors. Open **Admin Menu, Third Party Payor** to define your third-party payors.

Note: A third-party payor refers to an account. You must set up the account before you can create a third-party payor.

Fastpath: Refer to *A Promise To Pay May Reference A 3rd Party Payor* for more information.

Description of Page

The following fields display for each third party payor:

Third Party Payor Provide a meaning id for the third-party payor that can be easily recognized when setting up a promise to pay.

Description A meaningful description of the payor.

Account ID The account that is used for this payor. It is this account that is tracked as the "payee" of any payments made towards a third party payor's promise to pay.

Active Check this box if the payor is currently available to participate in promise to pay.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_THRD_PTY*.

Promise To Pay Cancel Reason

The **Promise To Pay Cancel Reason** screen allows you to view, edit, add and delete a promise to pay cancel reason. This screen consists of the following zones:

- *Promise To Pay Cancel Reason* on page 306

Promise To Pay Cancel Reason

The **Promise To Pay Cancel Reason** zone allows you to view, edit, add and delete a promise to pay cancel reason. This zone consists of the following fields:

Column Name	Column Description
Cancel Reason	Displays the promise to pay cancel reason.
Description	Displays the description of the promise to pay cancel reason.

In addition, this zone contains the following buttons:

Button Name	Button Description
Edit	Used to edit the promise to pay cancel reason.
Delete	Used to delete the promise to pay cancel reason.

You can add a new promise to pay cancel reason by clicking the **Add** link in the upper right corner of this zone.

Defining a Promise To Pay Cancel Reason

Procedure

To define a promise to pay cancel reason:

1. Click the **Menu** link in the **Actions/Navigation** area.

A list appears.

2. Select the **Admin Menu** option from the list.
3. From the **Admin Menu**, select **P** and then click **Promise To Pay Cancel Reason**.

   The **Promise To Pay Cancel Reason** screen appears.
4. Click the **Add** link in the upper right corner of the **Promise To Pay Cancel Reason** zone.

   The **Add/Edit Promise To Pay Cancel Reason** screen appears. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Cancel Reason	Used to specify the promise to pay cancel reason.	Yes
Description	Used to specify the description for the promise to pay cancel reason.	Yes

5. Enter the required details.
6. Click **Save**.

   The promise to pay cancel reason is defined.

Related Topics

For more information on...	See...
Promise To Pay Cancel Reason screen	*Promise To Pay Cancel Reason* on page 306
How to edit a promise to pay cancel reason	*Editing a Promise To Pay Cancel Reason* on page 307
How to delete a promise to pay cancel reason	*Deleting a Promise To Pay Cancel Reason* on page 308

Editing a Promise To Pay Cancel Reason

Procedure

To edit a promise to pay cancel reason:

1. Click the **Menu** link in the **Actions/Navigation** area.

   A list appears.
2. Select the **Admin Menu** option from the list.
3. From the **Admin Menu**, select **P** and then click **Promise To Pay Cancel Reason**.

   The **Promise To Pay Cancel Reason** screen appears.
4. Click the **Edit** (✎) icon in the **Edit** column corresponding to the cancel reason which you want to edit.

   The **Add/Edit Promise To Pay Cancel Reason** screen appears. It contains the following fields:

Field Name	Field Description	Mandatory (Yes or No)
Cancel Reason	Used to specify the promise to pay cancel reason.	Yes
Description	Used to specify the description for the promise to pay cancel reason.	Yes

5. Modify the required details.
6. Click **Save**.

   The changes made to the promise to pay cancel reason are saved.

Related Topics

For more information on...	See...
Promise To Pay Cancel Reason screen	*Promise To Pay Cancel Reason* on page 306
How to define a promise to pay cancel reason	*Defining a Promise To Pay Cancel Reason* on page 306
How to delete a promise to pay cancel reason	*Deleting a Promise To Pay Cancel Reason* on page 308

Deleting a Promise To Pay Cancel Reason

Procedure

To delete a promise to pay cancel reason:

1. Click the **Menu** link in the **Actions/Navigation** area.

   A list appears.
2. Select the **Admin Menu** option from the list.
3. From the **Admin Menu**, select **P** and then click **Promise To Pay Cancel Reason**.

   The **Promise To Pay Cancel Reason** screen appears.
4. Click the **Delete** ( ) icon in the **Delete** column corresponding to the cancel reason which you want to delete.

   A message appears confirming whether you want to delete the promise to pay cancel reason.
5. Click **OK**.

   The promise to pay cancel reason is deleted.

Related Topics

For more information on...	See...
Promise To Pay Cancel Reason screen	*Promise To Pay Cancel Reason* on page 306
How to define a promise to pay cancel reason	*Defining a Promise To Pay Cancel Reason* on page 306
How to edit a promise to pay cancel reason	*Editing a Promise To Pay Cancel Reason* on page 307

Creating Collection & Write-Off Procedures

Your collection procedures define how your organization collects overdue debt. Your write-off procedures define how your organization writes off finaled debt. In this section, we describe how to set up the data that controls these procedures.

Fastpath: For more information about collection and write-off procedures, see *The Big Picture Of Credit & Collections (C&C)*.

Caution: There are innumerable ways to design your collection and write-off procedures. Some designs will result in easy long-term maintenance, others will result in maintenance headaches. In this section, we provide information to help you understand the ramifications of the various options. Before you set up your production collection and write-off procedures, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

Designing Your Collection Procedures

The design of your collection procedures is an iterative process. Over time, you will develop intuitive skills that will allow you to skip some iterations. However, when you're starting out, we recommend you use the following matrix as your guide. When the matrix is complete, you're ready to set up the collection process control tables.

Account's Collection Class **Contract's Debt Class**	

The topics discussed below will gradually complete this matrix using a simple case study.

Fastpath: For more information about how the information in this matrix is used to monitor your customers' debt, refer to *Different Collection Criteria For Different Customers And Different Debt*.

Designing Your Debt Classes

Multiple debt classes are needed when you have different collection procedures for different types of contracts. If all contract debt is collected the same way, then you'll have just one debt class (call it `Generic`). However, if you're like many organizations, you will have multiple debt classes. The following points will help you understand why:

- If you levy deposits, you will probably have a deposit debt class. Why? Because you probably respond differently if the customer doesn't pay their deposit.
- If you allow customers to make payments on non-billed budgets, you will probably have a non-billed budget debt class. Why? Because you probably respond differently if the customer doesn't pay their non-billed budget (e.g., you may decide to expire the non-billed budget but not affect their other service since the non-billed budget is a way for customers to prepay for upcoming bills).
- If you write-off uncollectable debt, you will need another debt class for write-off contracts. Why? Because when you write-off debt in the system, you transfer the uncollectable debt from the original contract(s) to a write-off contract. The write-off contract holds this debt forever (or until it is paid). You need to use a different debt class for the write-off contracts because they aren't subject to collection criteria.
- If you use the system to charge your organization's company usage, you'll need another debt class (we refer to it as the "N/A" debt class below). Why? Because all contracts must have a debt class, even those that will never have debt.

Account's Collection Class **Contract's Debt Class**		
Deposit		
Non-Billed Budget		

Write Off		
N/A		

Designing Your Collection Classes

Multiple collection classes are needed when *any* debt class has different collection rules depending on the type of customer. If all customers within all debt classes are collected the same way, then you'll just have a single collection class (call it `Generic`). However, if you're like many organizations, you will have multiple collection classes.

Consider unregulated debt. For specific financial service industry customers, you probably don't worry until they owe you more than, say, $100 after 20 days. For other financial industry customers, you probably don't worry until they owe you more than, say, $5 after 20 days. In this situation, you will have at least two collection classes: one for specific financial service industry customers, the other for other financial industry customers.

Designing Collection Class Controls

At this point we have the rows and columns defined in our matrix. Now it's time to work on the individual cells.

Each cell should have a "collection class control" that defines its collection criteria and what to do if the criteria are violated. If a cell doesn't have a collection class control, this means you don't have any debt associated with that combination of collection class and debt class. So, we'll mark each cell without debt with "N/A".

Account's Collection Class **Contract's Debt Class**	Residential	Commercial/Industrial
Deposit		
Write Off		
N/A		

Next, we'll mark each cell for debt classes whose debt isn't collectable (i.e., the write-off and N/A debt classes).

Account's Collection Class **Contract's Debt Class**	Residential	Commercial/Industrial
Deposit		
Write Off	N/A	N/A
N/A	N/A	N/A

Note: If the Account Debt Monitor encounters debt associated with a non-existent collection class control, it will issue an error.

Determining the collection criteria in each remaining cell can be straightforward or complicated; it depends on how your organization works. Our case study assumes the following

- For deposit debt (regardless of collection class) we have multiple criteria:
    - Highest priority. If the customer is more than $5 in arrears by more than 50 days, kick off the Deposit Severely Overdue collection process. We'll talk more about this collection process later.
    - Lower priority. If the customer is more than $15 in arrears by more than 20 days, kick off the Deposit collection process. We'll talk more about this collection process later.

Given the above, our matrix will look as follows:

Account's Collection Class **Contract's Debt Class**	Residential	Commercial/Industrial
Deposit	Highest priority: If > $5 is older than 50 days, start Deposit Severely Overdue collection process. Lower priority: If > $15 is older than 20 days, start Deposit collection process.	Highest priority: If > $5 is older than 50 days, start Deposit Severely Overdue collection process. Lower priority: If > $15 is older than 20 days, start Deposit collection process.
Write Off	N/A	N/A
N/A	N/A	N/A

Designing Your Collection Class Control Overrides

Caution: Your collection needs may not require any overrides for your collection class control matrix and therefore this section may not be relevant.

The following matrix will help you design your collection class overrides. When the matrix is complete, you're ready to set up the collection class control tables.

Notice that the matrix has two dimensions: one is dependent on collection condition algorithms; the other is dependent on the collection class controls designed in the previous section. Collection condition algorithms are confusing. Think of them as optional conditions that, if met, will subject the collection class control's debt to different collection criteria.

Each cell in the matrix contains the collection criteria that will be applied to the account's debt when the collection condition is met (i.e., the same type of criteria - dollars and days and collection process - are defined in each cell).

We label the first collection condition as the `Default`. The collection criteria associated with this column will be used to analyze an account's debt when none of the other conditions applies. We'll start by indicating the `Default` collection criteria (this was defined in the previous section).

Account's Collection Class **Contract's Debt Class**	Default	Credit Rating < Threshold
Residential Charitable Contribution	See default collection criteria defined in previous section.	
Residential Regulated	See default collection criteria defined in previous section.	
Residential Unregulated	See default collection criteria defined in previous section.	
Commercial-Industrial Unregulated	See default collection criteria defined in previous section.	
Residential Deposit	See default collection criteria defined in previous section.	
Commercial-Industrial Deposit	See default collection criteria defined in previous section.	

If a different collection process OR criteria should be used when other conditions are met, you should indicate such by defining the collection criteria in the cell. For example, if we assume that all insurance debt has a different collection process when the account's credit score is less than the threshold credit rating on the installation record, our matrix will look as follows:

Account's Collection Class **Contract's Debt Class**	Default	Credit Rating < Threshold
Residential Charitable Contribution	See default collection criteria defined in previous section.	
Residential Regulated	See default collection criteria defined in previous section.	
Residential Unregulated	See default collection criteria defined in previous section.	Override: If Credit Rating is lower than the installation threshold: If > $5 is older than 15 days, start Risky Unregulated collection process.
Commercial-Industrial Unregulated	See default collection criteria defined in previous section.	
Residential Deposit	See default collection criteria defined in previous section.	
Commercial-Industrial Deposit	See default collection criteria defined in previous section.	

Once the matrix is complete, you're ready to design your collection process and collection event types.

Note: The collection conditions are limited by your imagination (and business requirements). We have provided the collection conditions you see above as an example; we don't expect you'll be able to use the exact conditions we supply. Your conditions will be based on any number of factors. For example, if you have different collection criteria that apply during winter months, you should add a new collection condition (called `Winter Season`). Or if you have different criteria based on years of service, you could have another condition.

Note: New collection conditions may require programming. See *How To Add A New Algorithm* for more information.

Designing Collection Process Templates & Collection Event Types

The following table shows the collection process templates referenced in the previous section's matrix. Adjacent to each process are its events and an indication of when they are triggered.

Collection Process Template	**Collection Event Template**	**Triggered X Days From Start Of Collection Process**
Large Overdue Debt	Large debt courtesy reminder letter	0
	To Do for large overdue debt	3
Deposit	Deposit reminder	0
Deposit Severely Overdue	Create To Do entry	0

If we extract each unique event type from the above table, we end up with the following:

Collection Event Type	Event Type
courtesy reminder letter	Send Letter - CHARIT REMIN
2 nd notice letter	Send Letter - REGUL 2 nd
Large debt courtesy reminder	Send Letter - LARGE REMIN
Risky debt courtesy reminder	Send Letter - RISKY REMIN
To Do for large overdue debt	Issue To Do
Large debt 2 nd notice letter	Send Letter - LARGE 2 nd
courtesy reminder letter	Send Letter - UNREG REMIN
2 nd notice letter	Send Letter - UNREG 2 nd
Deposit reminder	Send Letter - DEPOS REMIN
To Do for deposit severely overdue	Issue To Do

Now you're (almost) ready to set up your collection procedures.

Defining Cancellation Process Auto Cancellation Criteria

The topics in the section *How Are Collection Processes Cancelled* describe the two algorithms that play a part in the cancellation of a collection process. It also describes when to use what type of algorithm. Please read this section and then set up the appropriate cancellation criteria on your *Debt Classes*, and optionally, on your *Collection Process Templates*.

Setting Up Collection Procedures

In the previous section, *Designing Your Collection Procedures*, we presented a case study that illustrated a mythical organization's collection procedures. In this section, we'll explain how to set up the control tables to implement these procedures:

Setting Up Collection Event Types

Collection event types control what is done by a given collection event. Open **Admin Menu, Collection Event Type** to define your collection event types.

Description of Page

Enter a unique **Collection Event Type** and **Description** for the collection event type.

Enter the **Collection Event Type**. Permissible values are: `Affect Credit Rating/Cash-Only`, `Cancel Budget`, `Generic Algorithm`, `Send Letter`, `Create To Do Entry`. The following discussion describes the parameters that must be defined for each type of collection event.

The `Affect Credit Rating/Cash-Only` collection event type causes a credit rating demerit record to be linked to the account. This record is constructed using the following **Parameters**:

- Use **Credit Rating Points** to define this event's affect on the account's credit rating. This should be a negative number. An account's credit rating is equal to the start credit rating amount defined on the installation record plus the sum of credit rating demerits that are currently in effect. When an account's credit rating is less than the credit rating threshold defined on the installation record, the account's credit rating is displayed as an alert on Control Central.
- Use **Cash-Only Points** to define this event's affect on the account's cash-only score. This should be a positive number. When an account's cash-only score exceeds the cash-only threshold score defined on the installation record, the account is flagged as cash-only during payment processing and on Control Central.

- Use **Credit Rating Months** to define the length of time the demerit remains in effect. This information is used to define the effective period of the credit rating demerit record.

Fastpath: For more information, refer to *Account - Credit Rating* .

The `Send Letter` collection event type causes a customer contact to be generated that, in turn, generates a letter. Enter the following **Parameters** for this type of event:

- Select the **Contact Class** used to categorize the customer contact.
- Use **Contact Type** to define the type of customer contact to create. The type of customer contact controls the type of letter that is generated.

Note: Letter creation is triggered via a customer contact. You must set up a customer contact type for each type of letter you generate. You specify the necessary customer contact type on the collection event. Refer to *Setting Up Letter Templates* for more information.

The `Cancel Budget` collection event type cancel an account's budget plan (if the account is on such a plan). When a budget plan is cancelled, adjustments are issued to synchronize every contract's current balance with its payoff balance and each applicable contract's recurring charge amount (i.e., budget amount) is set to zero.

The `Generic Algorithm` collection event type causes the algorithm defined in the **Collection Event Alg** to be executed. You use this type of algorithm when the standard types of collection events won't do what you need done. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines the algorithm that will be called when events of this type of activated. Click *here* to see the algorithm types available for this plug-in spot.

The `Create To Do Entry` collection event type causes a To Do entry to be issued. A good example of where this is used is when the collection event requires that the customer be called on the phone. Refer to *The Big Picture of To Do Entries* for more information about To Do entries (refer to the To Do type `TD-CEVT` for the type of To Do entry that's created).

Enter a **Long Description** to fully describe the collection event type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_COLL_EVT_TYP*.

Setting Up Collection Process Templates

Collection process templates define the collection events that will be executed when a collection criteria rule is violated. Open **Admin Menu, Collection Process Template** to define your collection process templates.

Description of Page

Enter a unique **Collection Process Template** and **Description** for the collection process template.

Select a **Cancel Criteria Algorithm** if your organization allows individual contracts to be "removed" from a collection process regardless of the debt associated with all contracts in the debt class. In other words, if your cancel criteria are based on the debt associated with ALL contracts in a debt class DO NOT SPECIFY THIS ALGORITHM. If this algorithm is specified, it is executed by the collection process monitor when it detects that a credit has been applied to a contract linked to an active collection process. This algorithm will indicate if the specific contract that has been credited no longer has debt that warrants a collection process. Refer to *How Are Collection Processes Cancelled* for more information. If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to *Setting Up Algorithms*). On this algorithm, reference an Algorithm Type that "removes" a contract from a collection processes if the contract's debt so warrants. Click *here* to see the algorithm types available for this system event.

The **Response** grid contains an entry for every collection event that will be created when a collection process that references this template is created. The following information must be defined for each event:

Event Sequence Sequence controls the order in which the collection event types appear under the collection process template. The sequence number is system-assigned and cannot be changed. If you have to insert a collection event type between two existing templates, you'll have to remove the latter events, insert the new event, and then re-specify the removed events.

Collection Event Type Specify the type of collection event to be generated.

Days After Process Creation Specify the number of days after the creation of the collection process that the related collection event will be triggered. Refer to *Calendar vs Work Days* for a description of how this system uses this information to set the trigger date on the respective collection events.

Fastpath: For more information about collection event types, see *Setting Up Collection Event Types* . For more information about trigger dates, see *Collection Event Trigger Date*.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_COLL_PROC_TM*.

Setting Up Collection Classes

Every account has a collection class. This class is one of several fields that control the collection method applied to the account's debt. Open **Admin Menu, Collection Class** to define your collection classes.

Fastpath: For more information about collection classes, see *Designing Your Collection Classes*.

Description of Page

Enter a unique **Collection Class** code and **Description** for each collection class.

Indicate which method is used to monitor the member accounts' unpaid debt:

- If you practice *balance-forward accounting* for accounts belonging to this collection class, select `Collection & Write-Off`. This method of collection is described throughout this chapter.
- If you practice *open-item accounting* for accounts belonging to this collection class, select `Overdue`. This method of collection is described under *Defining Overdue Processing Options*.
- If accounts belonging to this collection class are not subject to either of the above collection methods, select `Not Eligible For Collection`. Please be aware that these accounts will NOT be reviewed for overdue debt.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_COLL_CL*.

Setting Up Debt Classes

Every contract type has a debt class. This class is one of several fields that control the collection criteria applied to the contract's debt. Open **Admin Menu, Debt Class** to define your debt classes.

Fastpath: For more information about debt classes, see *Designing Your Debt Classes*.

Description of Page

Enter a unique **Debt Class** and **Description** for the debt class.

Turn on **Eligible for Collection** if contracts belonging to this debt class have their debt monitored by the collection process. This should only be turned off if this debt cannot be collected, e.g., write-off debt.

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

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to an arbitrary number as only one algorithm for each system event is allowed in this case.

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

You can define algorithms for the following **System Events**:

System Event	Optional / Required	Description
`Collection Process Cancellation Rule`	Required if debt class is eligible for collection	This algorithm determines if a collection process can be canceled, and if so, it cancels it. Refer to *How Are Collection Processes Cancelled* for more information. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Override Arrears Due To Promise To Pay`	Required if you use promise to pay	This algorithm is called to temporarily override a customer's arrears using a promise to pay's scheduled payments when the system looks at an account's debt from a credit & collections perspective (i.e., the ADM calls this algorithm before it subjects a customer's debt to the collection criteria and when a promise to pay is created). It does not actually change any data, but overlays the current arrears with the promise to pay scheduled payments. This algorithm is also called by the above algorithms when a promise to pay is created in order to evaluate if the scheduled payments actually cover the arrears (if so, the collection processes are cancelled). It is also called periodically by the ADM in order to establish if the current state of the promise to pay still covers the arrears. Refer to *How Promise To Pay Affect The ADM*. Click *here* to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_DEBT_CL*.

Setting Up Collection Class Controls

The topics in this section describe the windows on which you set up your collection class control information.

Caution: The information in this page is what controls how the system analyzes your customer's debt. The flexibility of this control information provides you with almost unlimited options. This is very powerful, but it requires careful analysis. Refer to *Designing Your Collection Procedures* **for more information**.

Collection Class Control - Main Information

The information on this transaction defines the conditions that will be checked by the *Account Debt Monitor* when it checks if an account has violated your debt criteria.

Open **Admin Menu, Collection Class Control** to define this information.

Fastpath: For more information about collection class control, refer to *Designing Collection Class Controls*.

Description of Page

Enter a unique **Collection Class Control** code and **Description** for the collection class control (CCC).

Enter the **Division** to which the CCC's criteria applies.

Enter the **Collection Class** to which the CCC's criteria applies.

Enter the **Debt Class** to which the CCC's criteria applies.

Enter the **Currency Code** in which the CCC's criteria are denominated.

Use **Long Description** to further describe the CCC.

Fastpath: The information in the following grid is not intuitively obvious. Refer to *Designing Collection Class Controls* and *Designing Your Collection Class Control Overrides* for more information.

The grid which follows contains the conditions that are checked by the *Account Debt Monitor* (ADM) to determine the type of criteria (defined on the next tab) that will be applied against an account's debt. In other words - the ADM will check each condition (from highest to lowest **Priority**). The first condition that returns a value of true will cause the system to compare the account's debt against the debt criteria defined on the next tab.

Multiple conditions may be defined if different conditions result in a different type of debt thresholds (or a different type of collection process). The following fields are required for each condition:

Collection Condition Priority The priority controls the order in which the ADM checks if a collection condition applies (the lower the number, the higher the priority). Higher priorities are checked before lower priorities.

Note: The values for this field are customizable using the Lookup table. This field name is COLL_CAT_PRIO_FLG.

Condition Algorithm Define the algorithm used to check if an account should be subject to the collection criteria defined on the next tab. If the algorithm returns a value of true (i.e., the condition is met), the ADM will compare the account's debt against the **Debt Criteria** (defined on the next tab) and start a collection process if the account has debt that violates these criteria.

You must have at least one collection condition; otherwise the system will not have criteria against which to compare the account's debt. This entry should have the lowest priority code and reference the "default" algorithm. If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to *Setting Up Algorithms*). On this algorithm, reference an Algorithm Type that references the "default" collection condition algorithm type (*COLL COND DF*).

If you have other conditions that should be checked *before* the default condition, you must create an entry for each in this grid. Each entry should have a priority consistent with your business requirements (and this priority should be higher than the default condition's priority). In addition, you should reference an algorithm that contains the conditions that will be checked to determine if the account should be subject to the debt criteria (defined on the next tab). The system is supplied with many additional algorithm types. In order to take advantage of them, you will need to create an algorithm (refer to *Setting Up Algorithms*). On this algorithm, reference an Algorithm Type that references one of the collection condition algorithm types. Click *here* to see the algorithm types available for this system event.

Where Used

Collection class controls contain the data that controls the Account Debt Monitor. Refer to *How Does The Account Debt Monitor Work?* for more information.

Collection Class Control - Debt Criteria

The information on this page defines the debt and age thresholds used by the *Account Debt Monitor* when it checks if an account has violated your acceptable levels of debt. Open **Admin Menu, Collection Class Control** and use the **Debt Criteria** tab to define this information.

Note: The information on this page is not intuitively obvious. Refer to *Designing Collection Class Controls* and *Designing Your Collection Class Control Overrides* for more information.

Description of Page

The **Debt Criteria** scroll contains an entry for each collection criteria algorithm defined on the **Main** tab. The following information appears

The **Collection Condition Priority** controls the order in which the Account Debt Monitor (ADM) checks if a collection condition applies. Higher priorities are checked before lower priorities.

The **Condition Algorithm** is called by the ADM to determine which collection criteria should be applied to the account's debt. If this algorithm returns a value of true (i.e., the condition is met), the ADM will compare the account's debt against the **Debt Criteria** defined below. If the account violates any criteria, a collection process will be started (using the respective **Collection Process Template**).

The grid that follows contains the debt age and amount of debt that must be violated by the account in order for the ADM to create a collection process template. The following fields should be defined:

Arrears Priority Priority controls the order in which the arrears criteria will be checked by the Account Debt Monitor (the lower the number, the higher the priority). The first criteria, if any, that is met will cause a collection process to be created (using the **Collection Process Template**).

Note: The values for this field are customizable using the Lookup table. This field name is CRIT_PRIO_FLG. Be aware that this field is used for multiple tables: *Collection Class Control*, *Write Off Control* and *Workflow Process Profiles*.

Collection Process Template If the Account Debt Monitor determines that the account's debt violates the corresponding criteria, it creates a collection process using the specified collection process template.

Arrears Amount When the Account Debt Monitor checks an account's debt, it determines if the account has debt older than "> Number of Days" (the next field) AND the debt exceeds "> Arrears Amount". If so, a collection process is started.

Days When the Account Debt Monitor checks an account's debt, it determines if the account has debt older than **Days** AND the debt exceeds **Arrears Amount**. If so, a collection process is started.

Where Used

Collection class controls contains the data that controls the *Account Debt Monitor*.

Designing Your Write-Off Procedures

The design of your write-off procedures is relatively straightforward. Simply follow the instructions in the following topics.

Designing Your Write-Off Debt Classes

Multiple write-off debt classes are needed when you have different write-off procedures for different types of contracts. If all contract debt is written-off the same way, then you'll have just one write-off debt class (call it `Generic`). However, if you're like many organizations, you will have multiple write-off debt classes. The following points will help you understand why:

- If you bill for 3 rd parties, you probably have different write-off debt classes for the 3 rd party contracts. Why? Because you probably treat 3 rd party uncollectable debt differently from your own debt.
- You will need a separate write-off debt class for contracts whose debt cannot be written off. Why? Because there is a switch on the write-off debt class control table that controls if contracts in the write-off debt class are eligible for write-off processing. Given that you will have some contracts that hold debt that aren't eligible for write-off processing (e.g., contracts that hold written-off debt and contracts that overpayments), you will need at least one other write-off debt class.
- If you use the system to calculate charges for your organization's company usage, you'll need another write-off debt class (we refer to it as the "N/A" write-off debt class below). Why? Because all contracts must have a write-off debt class, even those that will never have debt.

Account's Collection Class Contract's Write-Off Debt Class		
Normal W/O		
N/A		

Designing Write-Off Controls

Set up a matrix using the collection classes you designed when you were designing your collection procedures (*Designing Your Collection Procedures*).

Account's Collection Class Contract's Write-Off Debt Class	Residential	Commercial/Industrial
Normal Write Off		
N/A		

Each cell should have a "write-off control" that defines what to do when the system detects finaled debt that hasn't been paid. This is true even of the "N/A" write-off debt class. Why? Because you may want the system to write-down these stopped contract's when they have a small balance. For example, if you have a write-off contract that subsequently receives a partial payment that leaves a small amount owing, you probably want the system to generate a write-down adjustment (so that the write-off contract will close). We'll initially fill in the matrix for the "N/A' write-off debt class.

Account's Collection Class Contract's Write-Off Debt Class	Residential	Commercial/Industrial
Normal Write Off		
N/A	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment.	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment.

Note: If the Write Off Monitor encounters debt associated with a non-defined collection class and write-off debt class, it will issue an error.

For each cell that isn't designated as N/A, you need to answer the following questions:

- Are you allowed to transfer debt to other non-closed contracts linked to the account? If so, you need to define the algorithm used to do such. Refer to *Setting Up Write-off Control* for more information about this algorithm.
- Are you allowed to write-down small amounts of debt (or small credits)? If so, you need to define the algorithm used to do such. Refer to *Setting Up Write-off Control* for more information about this algorithm.
- Should you refund credit balances with a check? If so, you need to define the algorithm to do such. Refer to *Setting Up Write-off Control* for more information about this algorithm.
- If debt remains after doing the above, how do you write it off (e.g., do you first refer the debt to a collection agency and only write it off after waiting 30 days)?

We'll fill in the above matrix with our assumptions:

Account's Collection Class Contract's Write-Off Debt Class	Residential	Commercial/Industrial
Normal Write-Off	Attempt to reduce the contract's balance to zero using the following methods: Attempt to transfer debt to another active contract linked to the account. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment. If the debt is <= $-1, create an A/P adjustment to refund the credit to the customer. If debt remains: Highest priority: If customer has a non-cash deposit, create the non-cash deposit write-off process. Otherwise, create the default write-off process for residential debt.	Attempt to reduce the contract's balance to zero using the following methods: Attempt to transfer debt to another active contract linked to the account. If the debt / credit is < $10 and > $-10, write down the debt using a write-down adjustment. If the debt / credit is <= $-10, create an A/P adjustment to refund the credit to the customer. If debt remains: Highest priority: If customer has a non-cash deposit, create the non-cash deposit write-off process. Otherwise, create the default write-off process for commercial debt.
N/A	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment. Because this debt class isn't eligible for further write-off processing, criteria used to process debt are not necessary.	Attempt to reduce the contract's balance to zero using the following methods: Synchronize current balance with payoff balance. If the debt is < $10 and > $-1, write down the debt using a write-down adjustment. Because this debt class isn't eligible for further write-off processing, criteria used to process debt are not necessary.

We can now use the information in the above matrix to design the necessary Write Off Process Templates and Write Off Event Types.

Designing Write Off Process Templates & Write Off Event Types

The following table shows the write-off process templates referenced in the previous section's matrix. Adjacent to each process are its events and an indication of when they are triggered.

Write Off Process Template	**Write Off Event Type**	**Triggered X Days From Start Of Write Off Process**
Residential	Refer to collection agency	0

Write Off Process Template	Write Off Event Type	Triggered X Days From Start Of Write Off Process
	Letter notifying customer of referral	0
	Cancel collection agency referral	60
	Write off	60
Non-Cash Deposit Exists	To Do for non-cash deposit redemption	0
	To Do to highlight unpaid contract(s) still exist (and they will be reconsidered for write-off processing)	10
Commercial	Refer to collection agency	0
	Letter notifying customer of referral	0
	To Do to check up on collection agency's efforts	30
	Cancel collection agency referral	60
	Write off	60

If we extract each unique event type from the above table, we end up with the following:

Write Off Event Type	Event Type
Notification of write-off referral	Send letter - Debt referred to a collection agency
Refer to collection agency	Refer to collection agency
Cancel collection agency referral	Cancel collection agency referral
Write off	Write off
To Do for non-cash deposit redemption	Generate To Do - Redeem non-cash deposit
To Do to highlight unpaid contract(s) still exist	Generate To Do - contract(s) linked to a non-cash deposit remain unpaid (and will be reconsidered for write-off processing)
To Do to check up on collection agency's efforts	Generate To Do - Check up on collection agency's efforts

And now you're ready to set up your write-off procedures.

Setting Up Write-Off Procedures

In the previous section, *Designing Your Write-Off Procedures* , we presented a case study that illustrated a mythical organization's write off procedures. In this section, we'll explain how to set up the control tables to implement these procedures:

Setting Up Write Off Debt Classes

Every contract type has a write-off debt class. This class is one of several fields that control the write off criteria applied to the contract's debt. Select **Admin Menu**, **Write Off Debt Class** to define your debt classes.

Fastpath: For more information about debt classes, see *Designing Your Write-Off Debt Classes* .

Panel controls

To modify a write-off debt class, simply move to a field and change its value. To add a new write-off debt class, click + to insert a row, then fill in the information for each field. The following fields display:

Write Off Debt Class Code The unique identifier of the write off debt class.

Eligible for Write Off Indicates if contracts belonging to this write off debt class are eligible for write-off processing. This should only be turned off if this debt cannot be written off, e.g., write off debt.

Description The description of the write off debt class.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_WO_DEBT_CL*.

Setting Up Write Off Event Types

Write-off event types control what is done by a given write-off event. Select **Admin Menu, Write Off Event Type** to define your write-off event types.

Description of Page

Enter a unique **Write Off Event Type Code** and **Description** for the write-off event type.

Enter the **Write Off Event Type**. Permissible values are: `Affect Credit Rating/Cash-Only`, `Cancel Agency Referral`, `Generic Algorithm`, `Refer to Agency`, `Send Letter`, `Create To Do Entry`, `Write Off using Distrib Code`, `Write Off using Contract Type`. The following discussion describes the parameters that must be defined for each type of write-off event.

The `Affect Credit Rating/Cash-Only` write-off event type causes a credit rating demerit record to be linked to the account. This record is constructed using the following **Parameters**:

- Use **Affect Credit Rating By** to define this event's affect on the account's credit rating. This should be a negative number. An account's credit rating is equal to the start credit rating amount defined on the installation record plus the sum of credit rating demerits that are currently in effect. When an account's credit rating is less than the credit rating threshold defined on the installation record, the account's credit rating is displayed as an alert on Control Central.
- Use **Affect Cash-Only Score By** to define this event's affect on the account's cash-only score. This should be a positive number. When an account's cash-only score exceeds the cash-only threshold score defined on the installation record, the account is flagged as cash-only during payment processing and on Control Central.
- Use **Months Affecting Credit Rating** to define the length of time the demerit remains in affect. This information is used to define the effective period of the credit rating demerit record.

Fastpath: For more information, refer to *Account - Credit Rating*.

The `Cancel Agency Referral` event type will cancel previous collection agency referrals. No parameters are needed for this type of event.

The `Generic Algorithm` write-off event type causes the algorithm defined in the **Generic Algorithm** to be executed. You use this type of algorithm when the standard types of write off events won't do what you need done. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines the algorithm that will be called when events of this type are activated. Click *here* to see the algorithm types available for this plug-in spot.

The `Refer to Agency` event type will refer the debt associated with the process' contract's to a collection agency. You must supply the **Agency Selection Algorithm** that is used to determine the collection agency associated with the referral. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines the collection agency to which bad debt should be referred. Click *here* to see the algorithm types available for this plug-in spot.

Note: Letters. You must set up a customer contact type for each type of letter you generate. You specify the necessary customer contact type on the write off event type. Refer to *Setting Up Letter Templates* for more information.

The `Send Letter` write-off event type causes a customer contact to be generated that, in turn, generates a letter. Enter the following parameters for this type of event:

- Select the **Contact Class** used to categorize the customer contact.
- Use **Contact Type** to define the type of customer contact to create. The type of customer contact controls the type of letter that is generated.

The `Create To Do Entry` write-off event type causes a To Do entry to be issued. Refer to *The Big Picture of To Do Entries* for more information about To Do entries (refer to the To Do type `TD-WOEVT` for the type of To Do entry that's created).

The `Write Off using Distrib Code` event type causes bad debt to be written off in accordance with the distribution codes associated with the financial transactions that caused the debt in the first place. Use this method if, for example, you want to write-off revenue differently than you write-off liabilities. When this type of event is activated, the system accumulates the distribution codes from GL details associated with recent financial transactions linked to each write-off contract. When the system has accumulated enough distribution codes (i.e., where the amount associated with the distribution code equals or exceeds the amount to write off), the debt will be transferred to a new or existing write-off contracts. The type of contracts to which the debt is transferred is defined on the distribution codes. Refer to *Setting Up Distribution Codes* for more information.

The `Write Off using Contract Type` event type causes all debt associated with the process's contract's to be transferred to a write-off contract linked to the account. Enter the following **Parameters** for this type of event:

- **Division / Contract Type** is the type of write-off contract to which the debt will be transferred. Note well,
  - The system will reuse an existing contract if an active contract of this type is already linked to the account; otherwise the system will create a new contract of this type.
  - The adjustment type used to set the offending contract's current balance equal to its payoff balance is defined on the write-offable contract Type. Refer to *Contract Type - Main Information* for more information.
  - The adjustment type used to transfer the delinquent debt to the write-off contract is defined on the write off contract type. Refer to *Contract Type - Detail* for more information.

Enter a **Comment** to fully describe the write-off event type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_WO_EVT_TYP*.

Setting Up Write Off Process Templates

Write-off process templates define the write-off events that will be executed when a write-off criteria rule is violated. Select **Admin Menu, Write Off Process Template** to define your write-off process templates.

Description of Page

Enter a unique **Write Off Process Template** code and **Description** for the write-off process template.

The rows in the following grid define the events that will be created when a write off process is created using this template. The following fields display:

Event Sequence Sequence controls the order in which the write-off event is executed. The sequence number is system assigned and cannot be changed. If you need to insert a write-off event between two existing events, you must remove the latter events, insert the new event, and then re-enter the removed events.

Write-off Event Type Code Specify the type of write-off event to be generated. The event type's description is displayed adjacent.

Days After Process Creation Specify the number of days after the creation of the write-off process that the related write-off event will be triggered. Refer to *Calendar vs Work Days* for a description of how this system uses this information to set the trigger date on the respective write-off events.

Fastpath: For more information about write-off event types, see *Setting Up Write Off Event Types* . For more information about trigger dates, see *Write-off Event Trigger Date* .

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_WO_PROC_TMPL*.

Setting Up Write Off Control

Write-off controls define how the system handles finaled, unpaid debt belonging to a given collection class and write off debt class.

Write Off Control - Main

Select **Admin Menu, Write Off Control, Main** to define basic information about a write-off control. After entering basic information, navigate to the **Criteria** tab to define the type of write-off process to start when given criteria are met.

Fastpath: For more information about write-off control, refer to *Designing Write-Off Controls* .

Panel controls

Enter a **Write Off Control** code and **Description** for the write-off control (WOC).

Enter the **Collection Class** to which the WOC applies.

Enter the **Write Off Debt Class Code** to which the WOC applies.

Enter general **Comments** to further describe the WOC.

Define the **Synch All Algorithm** used by the system to generate adjustments that cause current balance to equal payoff balance on the contracts to be written off. This type of algorithm is typically issued before you actually start a write-off process as current balance is meaningless at write-off time (the customer owes you the payoff balance). If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that synchronizes current and payoff balances. Click *here* to see the algorithm types available for this plug-in spot.

Define the **Debt Transfer Algorithm** used by the system when it attempts to transfer the unpaid debt to another active contract linked to the `stopped` contract's account. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).

- On this algorithm, reference an Algorithm Type that transfers unpaid balances. Click *here* to see the algorithm types available for this plug-in spot.

Define the **Write Down Algorithm** used by the system when it attempts to write-down small debt and/or credit balances. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that writes down small amounts. Click *here* to see the algorithm types available for this plug-in spot.

Define the **Credit Refund Algorithm** used by the system when it refunds a credit balance to a customer. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that refunds credit balances. Click *here* to see the algorithm types available for this plug-in spot.

Write Off Control - Criteria

Select **Admin Menu, Write Off Control, Criteria** to define the type of write-off process to start when given criteria are met.

Fastpath: The following information is not intuitively obvious. Refer to *Designing Write-Off Controls* for more information.

Panel controls

The information in the grid defines the write-off process to be executed for debt belonging to the previously defined collection class and write off debt class. The type of write-off process may differ depending on some condition. For example, you may have a different write-off process if the customer has a non-cash deposit. You must have at least one entry in this collection otherwise the system will not start a write-off process. This entry should have the lowest priority code and should reference a **Write Off Criteria Algorithm** that references the `WO CRIT DFLT` the algorithm type.

The following fields display:

Priority Priority controls the order in which the criteria will be checked by the Write Off Monitor (higher priorities are checked before lower priorities). The first criteria algorithm that is met (i.e., returns a value of *True*) will cause the associated write-off process to be initiated.

Note: The values for this field are customizable using the Lookup table. This field name is CRIT_PRIO_FLG. Be aware that this field is used for multiple tables: *Collection Class Control*, *Write Off Control* and *Workflow Process Profiles*.

Write Off Criteria Algorithm The Write Off Monitor checks if the condition defined by the W/O Condition Algorithm applies to the account whose debt is being analyzed. If a condition is met, a write-off process is created using the associated write-off process template.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines if a customer's bad debt should be processed using the associated **Write Off Process Template**. Click *here* to see the algorithm types available for this plug-in spot.

Note: Important! You must have at least one entry in this grid otherwise the system will not start a write-off process. This entry should have the lowest priority code and should reference a **W/O Criteria Algorithm** that references the *WO CRIT DFLT* algorithm type.

Write Off Process Template If the Write Off Monitor determines the condition defined by the w/o condition algorithm applies, a write-off process is created using the associated write-off process template.

Where Used

Write-off controls contain the data that controls the Write Off Monitor. Refer to *How Does The Write-Off Monitor Work?* for more information.

Setting Up Collection Agencies

You must set up a collection agency for each such organization to which you refer delinquent debt. To define a collection agency, select **Admin Menu, Collection Agency**.

Description of Page

Enter an easily recognizable **Collection Agency** code and **Description** for each collection agency.

A collection agency must be associated with a Person. Choose the **Person ID** of the organization from the prompt.

Fastpath: Information about how to set up persons is discussed in *Maintaining Persons* .

Turn on the **Active** switch if the collection agency is actively receiving referrals.

Specify the **Batch Control** that's used to route new and cancelled referrals to the collection agency. The batch control's description is displayed adjacent.

Where Used

Collection agencies get assigned to collection agency referrals when the collection agency referral background process executes. Refer to *How Do Collection Agency Referrals Work?* for more information.

Setting Up Feature Configuration

You must set up a *Feature Configuration* if you use the *champion / challenger* functionality.

The following describes the various **Option Types** that must be defined:

`Champion Template$Challenger Template$Percentage(1-100).` You need only set up options of this type if your implementation implements *Champion / Challenger* functionality. Options of this type are entered in the format `A$B$nnn` where A is the collection process template of the champion template, B is the collection process template of the challenger template, and C is the percent of the time that the system should create the challenger template. The collection monitor uses this option to override the champion collection process template X% of the time with the challenger template. You may enter any number of these options (but only one per Champion Template).

Chapter 6

Defining Cycles

Topics:

This chapter is dedicated to issues related to defining cycles in the system.

Defining Bill Cycles

Every account references a bill cycle. An account's bill cycle controls when it is billed.

In this section, we describe how to design and set up these cycles. In addition, we discuss how to set up bill period schedules. These are used to define the bill segment end date for contracts.

Note: Recommendation. We recommend reading *Bill Frequency - Bill Cycle vs Bill Segment Duration* before setting up this information.

The Big Picture Of Bill Cycles and Bill Periods

The topics in this section provide background information about a variety of bill cycle and bill period issues.

Bill Cycles

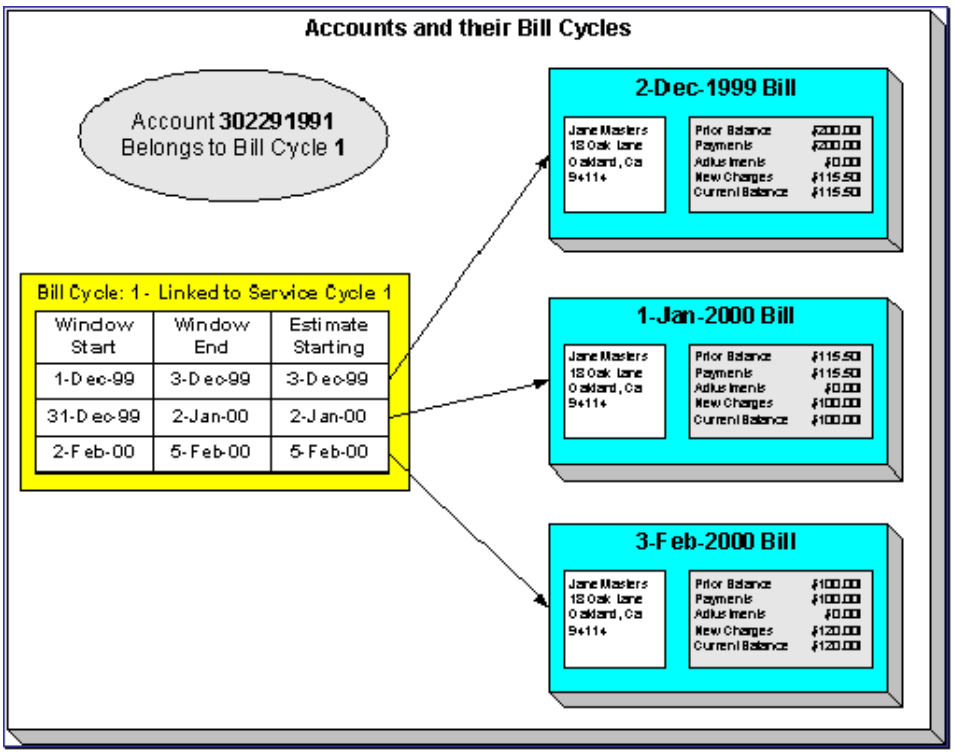
The topics in this section provide background information about a variety of bill cycle features.

The Cyclical Billing Process & Window Billing

The cyclical bill creation process creates most bills. This process works as follows:

- Every account references a bill cycle. The bill cycle's schedule controls WHEN the system attempts to create bills for the account.
- Every bill cycle has a bill cycle schedule that defines the dates when a cycle's accounts are to be billed. Rather than attempt to create bills on one evening, most bill cycles use a concept of "window billing" where the system attempts to produce bills for accounts over a few nights. The first night (i.e., the day the window opens) should be the earliest day, when the billable charges for the account can enter into the system. The last night (i.e., the day the window closes) should be the last possible day in which a billable charges can enter the system. Billable charges start date and end date should be within the boundary of window start date and window end date. This concept of window billing allows you to start billing accounts on the earliest possible day and then bill the stragglers over successive evenings. This results in much better cash flow.
- When the bill cycle creation process runs, it looks for bill cycles with open bill windows. It then attempts to create bills for the accounts in each such cycle. If a bill is created, it will send it out that evening. If a bill cannot be created, the system will create a bill in the "error" state with a message that can be analyzed by your billing staff. The next day, your staff can either correct the problem or not. When the bill cycle creation process next runs, it deletes all "error" bills and attempts to recreate them. It continues this throughout the bill window. If bills are in error at the end of the window, they will remain in this state until a user fixes them. If the bills are still in error when the cycle's next window opens, a billing error will be generated.

The following diagram should help clarify the above.



Fastpath: For more information about the how to control when bills are produced for a cycle, refer to *Bill Cycle - Bill Cycle Schedule*.

Designing Your Bill Cycle

The number of bill cycles is determined by the frequency that you bill your customers. So, for example, if you bill every month, you'll have 20 bill cycles - one for each bill day during the month. If you bill bimonthly, you'll have 40 bill cycles. If you bill quarterly, you'll have 60 bill cycles. Etc.

Keep in mind the following:

- You may need additional bill cycles if you allow customers to be billed off-cycle. For example, you could create a bill cycle called "Seniors" and link this to every senior citizen. You would set up this bill cycle's schedule to create bills shortly after social security checks are issued.
- You may need other bill cycles for customers of financial services.

Note: Important! An account's bill cycle should attempt to create bill segments at least as often as the shortest contract duration. For example, if an account has both monthly and quarterly contracts, the account should be placed on a monthly bill cycle. Refer to *Bill Frequency - Bill Cycle vs Bill Segment Duration* for more information.

How Does An Account Get Its Bill Cycle?

Most accounts are created behind-the-scenes when a user uses the "add account" option on *Person - Main Information* . An account created like this doesn't have a bill cycle. Rather, it sits in limbo awaiting the activation of its first contract. When a contract is activated, the system populates the account's bill cycle using the following algorithm:

- If an account's bill cycle is protected, the activation of a contract will not affect an account's bill cycle. Refer to *Protecting An Account's Bill Cycle* for more information.

Note: A To Do entry highlights accounts without a bill cycle. A To Do entry highlights those accounts that require a user to specify a bill cycle. This entry is generated for accounts without a bill cycle that have active contracts.

When the last contract linked to an account is `stopped`, the account's bill cycle will be changed so that the account will be final billed when billing next executes. Refer to *What Happens At Finalization Time?* for more information.

When a service point's service cycle is updated, and the account's bill cycle is not protected, the system automatically updates the account's bill cycle to be in sync with the service cycle.

Protecting An Account's Bill Cycle

Some times customer may want to change bill cycle . If you do not want the system to change an account's bill cycle when a contract is activated, you need to turn on the account's **protect bill cycle** flag. You would do this if a customer liked to be billed at the start of the billing month.

When the last contract for an account is stopped, the protect bill cycle flag is reset. This is to ensure that if the customer returns to start new service again, the bill cycle can be set based on *How Does An Account Get Its Bill Cycle*.

Designing Bill Periods

Bill periods are used by contracts whose bill end dates need to fall on strict dates. You need only set up this information if you have this type of contract.

Fastpath: Refer to *Ways To Control The End Date Of A Bill Segment* for more information.

Every bill period has a calendar that defines when bill segments are created for contracts that use the bill period. Examples of bill periods include:

- Quarterly Bill - Last Day Of Quarter
- Quarterly Future Bill - Last Day Of Quarter
- Monthly - 15 th Day Of Month.
- Monthly Future Bill - Last Day Of Month.

The Quarterly Bill - Last Day Of Quarter bill period would have a schedule that looked as follows:

Earliest Date On Which To Create A Bill Segment	**Bill End Date**
1-Oct-1998	30-Sep-1998
1-Jan-1999	31-Dec-1998
1-Apr-1999	31-Mar-1999
1-Jul-1999	30-Jun-1999
...	

The Quarterly Future Bill - Last Day Of Quarter bill period would have a schedule that looked as follows:

Earliest Date On Which To Create A Bill Segment	**Bill End Date**
15-Dec-1998	31-Mar-1999
15-Mar-1999	30-Jun-1999
...	

The remainder of this section provides examples using the above calendars.

The following example assumes the following:

- The contract starts on 18-Dec-1998.
- The contract's contract type references the Quarterly Future Bill - Last Day Of Quarter bill period.

The following table shows when bill segments will be produced (assuming the account's bill cycle attempt to create segments as soon as possible) for several bill periods:

Earliest Date Segment Will Be Produced	Bill Period
18-Dec-1998	18-Dec-1998 thru 31-Mar-1999
15-Mar-1999	1-Apr-1999 thru 30-Jun-1999
...	

The following example assumes the following:

- The contract starts on 18-Dec-1998.
- The contract's contract type references the Quarterly Bill - Last Day Of Quarter bill period.

The following table shows when bill segments will be produced (assuming the account's bill cycle attempts to create segments as soon as possible) for several bill periods:

Earliest Date Segment Will Be Produced	Bill Period
1-Jan-1999	18-Dec-1998 thru 31-Dec-1998
1-Apr-1999	1-Jan-1999 thru 31-Mar-1999
...	

Fastpath: Refer to *Setting Up Bill Periods* for information about how to define this information.

Setting Up Bill Cycles

An account references a bill cycle. The bill cycle defines when the account is billed and when the account's debt is checked to determine if it's overdue. To define a bill cycle and its bill cycle schedule, open **Admin Menu, Bill Cycle**.

Description of Page

Enter a unique **Bill Cycle** and **Description** for every bill cycle.

Use the **Bill Cycle Schedule** collection to define when bills are produced for the accounts in a given bill cycle. The following fields are required for each instance:

Window Start Date Specify the date on which the system should start trying to create bills for accounts in the cycle.

Window End Date Specify the last day on which the system will create bills for accounts in the cycle.

Accounting Date Specify the financial date associated with the bills' financial transaction. The accounting date defines the financial period(s) to which the bills will be booked in your general ledger.

Freeze and Complete Turn on this switch if the system should freeze and complete any bill that is created without errors. If this switch is turned off, all bills created by the billing process will be left in the unfinished state. You would only turn this switch off if you want to verify an entire bill run prior to freezing it (e.g., if you are introducing a new version of a rate). If you turn this off, you will need to return to this page after verifying a bill run and turn it back on for the customers to receive bills. When the system next runs, it deletes all unfrozen bills and recreates them as per the instructions on the bill cycle schedule.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_BILL_CYC*.

The batch bill creation process uses this schedule to define the bill cycles for which it should create bills.

Note: You can delete a bill cycle only if it is not associated to any division.

Setting Up Bill Periods

Some contract types reference a bill period. The bill period defines when the contract's bill segments are produced and the respective end date of each bill segment.

Fastpath: Refer to *Designing Bill Periods* for more information.

To define a bill period and the bill period schedule, open **Admin Menu, Bill Period**.

Description of Page

Enter a unique **Bill Period** and a **Description** for every bill period.

Use the **Bill Period Schedules** collection when the system should create bill segments for contracts that use a given bill period. It also defines the end date of each respective bill segment. The following fields are required:

Bill Date Specify the earliest date on which the system is allowed to create a bill segment for contracts using this bill period.

Bill Seg End Date Specify the end date of the bill segment. For future bills, this will be after the bill date. For retro bills, this will be before the bill date.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_BILL_PERIOD*.

This information is used by the bill segment creation process to determine the end date of contracts that use a bill period.

Defining Statement Cycles

If you have persons set up in the system to receive statements with financial information, you will need to assign them to a statement cycle and define a schedule for the statement cycle.

Refer to *The Big Picture of Complex Statements* for more informationabout statements.

The Big Picture Of Statement Cycles

A statement cycle has a similar purpose to that of a bill cycle. It controls when statements will be produced.

The Cyclical Statement Process

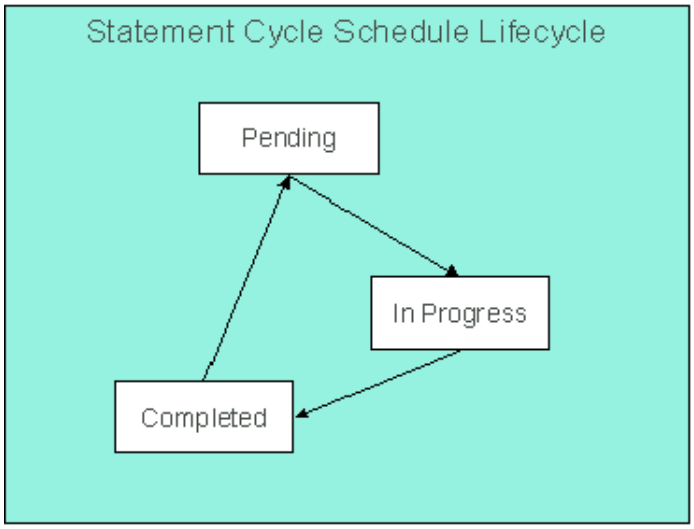
Persons who wish to receive statements will work with you to determine how often these statements should be produced. Some persons may want a monthly statement, some a quarterly and some annually. For each unique schedule that is designed for your various statement persons, you will set up a Statement Cycle and its schedule.

Designing Your Statement Cycles

The number of statement cycles is determined by a combination of the frequency that you will send statements to the statement persons and how many statement cycles you wish to manage within the same frequency.

So, for example, for all the statement persons who wish to receive a monthly statement, will you create only one monthly Statement Cycle so that all monthly statements are produced the same day? Or will you have several monthly statement cycles scheduled throughout the month? The answer will depend on the volume of statements being produced and on how you want to manage the statement production.

Lifecycle of a Statement Cycle Schedule



`Pending` The statement cycle schedule is added in this state. The Create Statements background process find records in this state to process on the appropriate date.

Fastpath: Refer to *Create Statements Background Process* for more information.

`In Progress` Records in this state are currently being processed by the Create Statements background process.

`Completed` Records in this state have already been processed by the Create Statements background process. If a problem occurs with the Create Statements background process and it needs to be rerun, simply change the status back to `pending` and rerun the process.

Setting Up Statement Cycles

A Statement Construct references a statement cycle. The statement cycle defines when the statement person will receive statements with financial information related to the accounts and contracts linked to the statement construct. To define a statement cycle and its statement cycle schedule, open **Admin Menu, Statement Cycle**.

Description of Page

Enter a unique **Statement Cycle** and **Description** for every statement cycle.

Use the **Statement Cycle Schedule** collection to define when statements are produced for the persons with statement construct records in the given statement cycle. The following fields are required for each instance:

Processing Date Specify the date on which the system should create statements for persons with statement construct records in the cycle.

Status Indicates the status of the cycle schedule. Refer to *Lifecycle of a Statement Cycle Schedule* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_STM_CYC*.

The batch statement creation process uses this schedule to determine which statement cycles for it should create statements for the statement construct records. Refer to *Create Statements Background Process* for more information.

Chapter 7

Defining Contract Types

Topics:

Every contract must reference a contract type. The contract type defines what you sell, how much you sell it for, to whom you sell it, how overdue debt is collected, and how sales will be booked in your general ledger.

Note: Perfect foresight. In a perfect world, the other control tables would have been set up with perfect foresight of setting up your contract types. In reality, setting up your contract types may invalidate some of your earlier decisions. Don't feel bad if this happens, some amount of iteration is natural.

Designing Contract Types For Contracts

The topics in this section provide guidelines describing how to design the contract types associated with your contracts.

Overpayment Segmentation

When a customer pays more than they owe, you must decide what to do with the excess money. The following points describe two possibilities:

- You could create a new contract to hold the excess (let's call it an overpayment contract). The credit would be transferred from this contract to the billable contracts when the next bill is completed. This means that all billable contracts have the same opportunity to receive the overpayment when they are billed in the future.
- You could amalgamate the excess payment on one of the existing, billable contracts. For example, if a customer has both banking and insurance service, the excess funds could be kept on either the banking or insurance contract. This would result in the following:
  - The contracts that do NOT receive the overpayment will have debt when they are next billed.
  - The contract that receives the overpayment could have its future debt offset by the overpayment (meaning that it could have a credit balance until the contract's future bill segments offset the overpayment amount).
- The above situation is not desirable unless the customer intentionally overpaid one contract. The first method (keeping the overpayment on a separate contract) obviates this potential problem. Obviously, if your organization sells a single service (and therefore your customers have a single contract) you would choose the second method.

You control which method is used by plugging in the appropriate `Overpayment Distribution` algorithm on each *Customer Class* (i.e., you can choose a different method for different customer classes). If you choose to hold overpayments on a separate contract, then you must set up an contract type as described in the following table:

Division/ Contract Type	Service Type	Distrib.Code	Eligible for Billing	Debt Class	Pay Seg Type	Do Not Overpay	One-time
CA/ OVERPAY	Financial Service	A/P - OVER	Not billed	N/A	Normal	No	Yes

Notice the following about the new overpayment contract type:

- It has an interesting distribution code. This is because when a payment segment is created for this type of contracts, the system must credit a liability (an overpayment is a liability).
- It's important to indicate that the overpayment contract is a one-time contract. Why? Because this means that the system will automatically close the contract when it's balance falls to zero (i.e., when all of the overpayment has been used to satisfy future bills).
- A bill segment type is not needed because the system never creates bill segments for such contracts (they exist only to hold excess credits).
- You may also want to turn on the alert message.

Caution: Important! You must plug-in a bill completion algorithm on this contract type. This bill completion algorithm will transfer the credit balance to the account's other contracts when the bill is completed. Refer to *The Credit Transfer Algorithm* for more information about this algorithm.

- You must also reference this overpayment contract type as the parameter value on your overpayment algorithm (this algorithm is plugged in on the desired customer classes). Refer to *Overpayment Algorithm* for more information about this algorithm.

Note: If overpayment means charitable contribution. Some organizations sponsor a program that works as follows - if a customer overpays a bill by a given amount (say $5), this amount is assumed to be a charitable contribution. If you have this requirement, you will create another contract type to hold a customer's charitable contributions. The funds will be credited to this contract by creating a new overpayment algorithm that is similar to the base package *Overpayment Algorithm*. This new algorithm will be very similar to the existing algorithm. The main difference will be that it will have to check if the overpayment amount is an exact value (say $5). If so, it will create a payment segment for the charitable contribution contract type; otherwise it will create a payment segment for the overpayment contract.

Write Off Segmentation

When you write off non-collectable debt, you transfer the receivable from a "normal" contract onto one or more write-off contracts. When the debt is transferred to a write-off contract, the distribution code on the "normal" contract is credited (typically an A/R GL account), and the distribution code on the write-off contract is debited.

You will almost always need a write-off contract whose distribution code is the write-off expense. However, you probably don't book all of the write-off amount to a write-off expense account. Why? Because the debt that you're writing off typically contains both revenue and liabilities. At write-off time, you want to book the written off revenue to a write-off expense account and you want to reduce the liabilities (you don't owe the liability if you don't get paid). This means you'll need another contract type for the liabilities. Refer to *The Ramifications of Write Offs in the General Ledger* for a complete explanation.

The following table contains the minimum number of contract types that you'll need to hold your write-offs.

Division/ Contract Type	Service Type	Distrib.Code	Bill Seg Type	Debt Class	Pay Seg Type	Do Not Overpay
CA/WO-STD	Other	EXP-W/O	Not billed	WO	Normal	Yes
CA/WO-LIA	Other	LIA-General	Not billed	WO	Normal	Yes

Notice the following about the new write-off contract types:

- They have interesting distribution codes. This is because when debt is transferred to these types of contracts, the system must debit either an expense account (i.e., write-off expense) or a liability account. It's important to note that in *The Ramifications of Write Offs in the General Ledger* we explain how this liability account may be overwritten with the liability account that was originally booked.
- Neither needs a bill segment type because the system never creates bill segments for such contracts (they exist only to hold uncollectable debt)
- Even though the debt is not collectable, it still has a debt class. Why? Because the system shows a customer's debt on many inquiries by debt class and it's important to show write-off debt on these queries.
- The combination of Payment Segment Type and Do Not Overpay are important. Refer to *The Ramifications of Write Offs in the General Ledger* for a complete explanation.

Note:

The adjustment type used to set the offending contract's current balance equal to its payoff balance is defined on each write-offable contract type. The adjustment type used to transfer the delinquent debt to the write-off contract is defined on the write-off contract type.

An Alternative. If you have a limited number of liability accounts, you may decide to have a separate write-off contract for each liability account. Doing this would proliferate the number of contracts created at write-off time. However, it would simplify the remittance of payment to the taxing authority if the reversed liability is ever paid.

Payment Arrangement Segmentation

If your organization allows customers to payoff outstanding debt using payment arrangements (e.g., current bill plus $X), you will need a new contract type for every debt class that can have a payment arrangement. If we assume you can have payment arrangements for both regulated and unregulated debt, then you'll need at least two more contract types (you may have more contract types if you need to segregate the payment arrangement receivable amount).

Division/ Contract Type	Service Type	Distrib.Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/PA-REGU	Other	A/R-ARRG	REGU	RECUR-AS	Amount to bill is `Not Allowed` Amount is `Required` Frequency is `Monthly` Recurring Amount Label is `Arrange Amount`
CA/PA-UNRE	Other	A/R-ARRG	UNRE	RECUR-AS	Amount to bill is `Not Allowed` Amount is `Required` Frequency is `Monthly` Recurring Amount Label is `Arrange Amount`

Notice the following about the new payment arrangement contract types:

- They have an interesting distribution code. This is because when funds are transferred to these types of contracts, the system must debit a receivable (i.e., payment arrangement receivable).
- They use an interesting bill segment type - RECUR-AS. This bill segment type was set up to create recurring charges that stop when the customer no longer has a payoff balance.
- Each new contract type references the debt class whose debt it will pay off. We are intentionally linking the payment arrangement debt to the same debt class as the regulated debt from which it originates. This way, the C&C process will consider the arrangement debt as the same as regulated debt and therefore perform the regulated collection.
- The recurring charge control information is set up as defined.

Fastpath: Refer to the Description of Page under *Contract Type - Billing* **for the definition** of the recurring charge attributes.

Merchandise Segmentation - Installment Billing

If your organization allows customers to purchase merchandise using an installment plan, you must create a contract type for this.

Note: No installments. If the customer must pay for the merchandise in one lump amount, you'd create an contract type.

Division/ Contract Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/ MERCH-I	Merch	A/R-MRCH	UNRE	RECUR-AS	Amount to bill is `Not Allowed` Amount is `Required` Frequency is `Monthly` Recurring Amount Label is `Install Amount`

Notice the following about the new merchandise contract type:

- It has a normal receivable distribution code.
- It uses an interesting bill segment type - RECUR-AS. This bill segment type was set up to create recurring charges that stop when the customer no longer has a payoff balance.
- The recurring charge control information is set up as defined.

Fastpath: Refer to the Description of Page under *Contract Type - Billing* **for the definition** of the recurring charge attributes.

Deposit Segmentation - Installment Billing

If your organization allows customers to pay deposits using an installment plan, you must create an contract type for this.

Note: No installments. If the customer must pay for the deposit in one lump amount, you'd create an contract type. Just make sure the adjustment that's levied to charge for the deposit amount doesn't affect payoff balance (when you bill a deposit, the customer doesn't really owe anything because it's not a true receivable from an accountant's perspective).

Division/ Contract Type	Service Type	Distrib. Code	Debt Class	Bill Seg Type	Recurring Charge Control Info
CA/DEP-I	Other	A/P-DEPO	DEP	RECURATB	Amount to bill is `Required` Amount is `Required` Frequency is `Monthly` Recurring Amount Label is `Install Amount`

Notice the following about the new deposit contract type:

- It has an interesting distribution code. This is because when a payment is distributed to these types of contracts, the system must credit a payable account (i.e., deposit payable) rather than a receivable account. Note well, we have assumed a receivable is not incurred when the bill segment for the deposit is created.
- It uses an interesting bill segment type - RECURATB. This bill segment type was set up to create recurring charges that stop when the system has billed the Total Amount to Bill.
- The debt class is interesting - DEP (for deposit). This is done so that past due deposit "debt" is treated separately from other types of debt.
- The recurring charge control information is set up as defined. Note well, the Amount to bill is `Required`.

Fastpath: Refer to the Description of Page under *Contract Type - Billing* **for the definition** of the recurring charge attributes.

Note: Bill messages on receipt of deposit in full. The base package includes a special FT Freeze algorithm that can be specified on deposit contract types. It recognizes when a deposit has been paid in full, and creates a bill messages to inform the customer. Refer to algorithm `DEP PIF MSG` in *Algorithm Types* for more information.

Billable Charge Segmentation

You create a billable charge whenever a customer should be charged for a service that occurs outside the normal course of business. You can also use billable charges to "pass through" other bill ready charges generated outside the system, by another application, or by a 3 rd party supplier.

A billable charge must reference a contract. This contract behaves just like any other contract:

- **Bill segments are created for the contract.** Whenever billing is performed for an account with billable charge contracts, the system creates a bill segment for each contract with unbilled charges. If multiple unbilled charges exist for a given contract, only one bill segment will be created and it will contain details about all of the billable charges.
- **Payments are distributed to the contract.** Payments made by an account are distributed to its billable charge contracts just like any other contract.
- **Overdue debt is monitored.** The credit and collections process monitors billable charge contracts for overdue debt and responds accordingly when overdue debt is detected.

Therefore, you must set up at least one contract type to hold your billable charge debt. You may have multiple charges based on billing frequencies, A/R booking, debt monitoring, etc. It's really up to you.

The easiest way to determine how many billable charge contract types you'll need is to define every conceivable billable charge (which you should have done when you designed your *billable charge templates*). Then ask yourself if they have the same billing and payment behavior, if so, you'll have one contract type. If not, you'll need one contract type for each combination.

We will assume your billable charges are all used to levy unusual one-off charges that can be collected in the same way, therefore we'll need one contract type.

Division/ Contract Type	Service Type	Distrib Code	Debt Class	Bill Seg Type	Billable Charge Templates	Rate
CA/ ONETIME	Other	A/R-OTH	OTH	BILLCHRG	TREETRIM DAMAGE	None
CA/ PASSTHRU	Banking	A/R-BK	BK	BILLCHRG	None	None
CA/ ADDON	Insurance	A/R-INS	INS	BILLCHRG	None	TAXES

Notice the following about the new *one time* contract type:

- It has a normal receivable distribution code.
- Its debt class is unregulated.
- It uses an interesting bill segment type - BILLCHRG. This bill segment type was set up to create bill segments using billable charges.
- It references the valid billable charge templates that can be used on this contract type.

Note: One Time Charge. The `ONE TIME` example shown above implies this contract type exists to hold one-time charges. Because of this, you should turn on the One Time Charge switch on the contract type so that contract's of this type are automatically closed when final payment is received. You don't have to do this because a customer could have a single billable charge contract that is perpetually active for pass through charges (i.e., it doesn't have a stop date). If you do this, the system will create a bill segment for this contract whenever it finds an unbilled billable charge linked to the contract.

Notice the following about the *pass through* contract type:

- It doesn't use the normal distribution code or debt class. This is done so that the debt and receivable can be tracked separately. If these charges were being pass through from another system, you might want to track these financial values separately.
- It still uses the normal bill segment type - BILLCHRG. From a billing perspective, there is no difference between this and the one time contract type.
- Templates are not relevant - these charges on not created on-line using templates, but are loaded via the *Billable Charge Upload Staging*.

Notice the following about the *add on charges* contract type:

- This is an example of bill-ready charges (similar to pass through) to which the system adds on other charges, for example, taxes.
- It still uses the normal bill segment type - BILLCHRG. From a billing perspective, there is no difference between this and the one time contract type.
- It also uses a Rate. In this case, the bill creation algorithm (specified on the bill segment type) will take any billable charge lines and attach them to a bill. In addition, these billable charges will include billable charge service quantities (SQs). These service quantities will also be swept onto the bill segment, and the Rate (TAXES in this example) will be applied. In order for taxes to be calculated, the billable charge SQs must

include the total taxable amount - the system is not able to apply the rate on top of the other billable charges. But, it can apply the tax rate to the SQs that are supplied.

- If the rate has SQ Rules, these will be applied as well.

Fastpath: For more information about billable charge templates, refer to *Setting Up Billable Charge Templates*.

Over/Under Cash Drawer Segmentation

In order to balance a tender control that is out-of-balance, your organization must set up an account with a contract whose contract type references the over/under expense account. You will probably only have one contract that references this contract type, but you still must have it if you remit funds via a cash drawer.

Fastpath: For more information about over/under processing, refer to *How To Get An Unbalanced Tender Control In Balance (Fixing Over/Under)*.

Division/ Contract Type	Service Type	Distrib.Code	Debt Class	Bill Seg Type
CA/OVR UNDR	Other	EXP-OV/UND	N/A	Not billed

Notice the following about the new contract type:

- It has an interesting distribution code. This is because when a payment segment is applied to this type of contract, the system must debit an expense account for under amounts (and credit it for over amounts).
- It doesn't need a bill segment type because the system never creates bill segments for such contracts (it only has over/under payment segments linked to it).
- It uses the N/A debt class because the credit and collections process should never consider debt associated with contracts of this type (because it's not really debt).

Payment Upload Error Segmentation

If the payment upload process detects an invalid account on a payment upload record, it will create a payment for the suspense contract defined on the upload process' tender source (see *Setting Up Tender Sources*). You should create a special contract type for this contract.

Fastpath: For more information about the payment upload process, refer to *Phase 3 - Create Payment Events, Tenders, Payments and Payment Segments*.

Division/ Contract Type	Service Type	Distrib.Code	Debt Class	Bill Seg Type
CA/SUSPENSE	Other	EXP-MISC	N/A	Not billed

Notice the following about the new contract type:

- It has an interesting distribution code. This code should probably be a suspense account. All payment segments that are created for this contract will eventually be transferred to a "real" contract and therefore this GL account's balance should be zero when no payments are in suspense.
- It doesn't need a bill segment type because the system never creates bill segments for such contracts (it only has invalid account payment segments linked to it).

- It uses the N/A debt class because the credit and collections process should never consider debt associated with contracts of this type (because it's not really debt).

Loan Segmentation

If you loan money to customers that is recouped using an amortization schedule, you need to set up an contract type for the loan contract.

Division/ Contract Type	Service Type	Distrib.Code	Loan A/ R Distrib. Code	Debt Class	Bill Seg Type
CA/LOAN	Other	A/R - LOAN	A/R - STLN	UNRE	LOAN

Fastpath: Refer to *Setting Up The System To Enable Loans* for more information.

Non-billed Budget Segmentation

If you allow your customers to pay set amounts at specified intervals (e.g. every two weeks), you need to set up contract types for non-billed budget contracts. Non-billed budgets are typically used when your company bills on an infrequent basis and you want to provide your customers with a mechanism to make smaller payments more frequently. You may implement two types of non-billed budgets monitored and unmonitored, each type requiring a different contract type. You may also implement different renewal options for non-billed budgets.

Division/ Contract Type	Service Type	Distrib. Code	Non Billed Budget Monitoring	Debt Class	Renewal	Bill Seg Type
CA/NBB-MON	Other	A/P - OVPY	Monitored	NBB	Optional	Not billed
CA/NBB-UNMON	Other	A/P - OVPY	Unmonitored	N/A	Optional	Not billed

Designing Contract Type For Other Segmentations

The earlier parts of this discussion described the most common factors that cause the creation of contract types. However, many obscure factors could cause the introduction of more contract types. In this section, we explain these more obscure factors.

Caution: We strongly recommend not being too pedantic when considering the factors described in this section. If you can only think of a few strange situations that would necessitate a contract type, think carefully before you introduce it. It's better to be a little less than perfect than end up with large number of obscure contract types.

Cash Distribution Segmentation

Every contract type has a payment segment type. The payment segment type defines the cash account to which the contract type's payments should be booked. If different contracts have different cash accounts, you will need to split the contract types accordingly.

Adjustment Profile Segmentation

Every contract type has one or more adjustment profiles. These profiles define the valid adjustment types that can be booked to the contract type's contracts. If different contracts within an contract type have different mixtures of valid adjustment types, you must split the contract types accordingly.

Late Payment Charge Segmentation

An option exists on contract type that causes the system to generate a late payment charge if payment is not received on time. If you don't levy late payment charges on all contracts, you will need to determine when you do and design your contract types accordingly.

In addition, if you levy late payment charges, the percentage levied and the algorithm that defines the amount of the outstanding balance subject to the charge is defined on the contract type.

Debt Classification Segmentation

Every contract type has a debt class. The debt class is used to categorize a contract's debt for the purpose of credit and collections (C&C) analysis. If a given contract type has different categories of debt from C&C's perspective, you must split the contract type.

Fastpath: For more information about debt class, refer to *Designing Your Collection Procedures*.

Note: Write Off Debt Class vs. Normal Debt Class. An contract type references both write off debt class and normal debt class. An contract type's write-off debt class controls the write-off rules imposed on contracts of a given type. An contract type's normal debt class controls the collection rules imposed on contracts of a given type. Refer to *Different Collection Criteria For Different Customers And Different Debt* for more information about collection rules. Refer to *Different Write-Off Criteria For Different Customers And Different Debt* for more information about write-off rules.

Allow Estimates Segmentation

Every contract Type has a switch that controls whether the system estimates consumption at billing time. If given contract type has different situations when the system should and should not estimate, you will have to split the contract type.

Note: Override Note. You can override the value of the contract type's estimation switch on an individual contract. This means that if only a few contracts don't abide by the contract type's estimation switch, you can change the switch value of these contracts.

Deposit Class Segmentation

Every contract type that exists to hold a cash deposit will reference a deposit class. The deposit class defines the business rules that control various functions including interest calculation and refund criteria. You will need multiple deposit contract types if any of the deposit class' rules / conditions differ for different types of deposits. For example, if residential customers use a different recommended deposit algorithm as compared to commercial customers, you'd need one contract type for residential deposits and another for commercial deposits (where the residential deposit contract type will reference the residential deposit class and the commercial deposit contract type will reference the commercial deposit class).

You will need additional deposit contract types if your customers can have multiple deposits where each deposit is restricted to a specific type of debt.

Fastpath: For more information about deposit class, refer to *Designing and Defining Deposit Classes*.

Processing Sequence Considerations

You may have customers with a complex collection of contracts such that the calculation for one bill segment relies on information calculated by another bill segment for the same account.

The billing processing sequence also controls the order of contracts in the following other processes:

- Execution of pre bill completion algorithms. The system processes each contract in the billing processing sequence order. Within each contract, the pre bill completion algorithms on its *contract type* are processed in the order of the algorithm's sequence.
- Execution of bill completion algorithms. The system processes each contract in the billing processing sequence order. Within each contract, the completion algorithms on its *contract type* are processed in the order of the algorithm's sequence.
- Interval Data Creation. For interval contracts linked to interval profiles with profile creation algorithms defined, the system processes each contract in the processing sequence order. Within each contract, the data creation algorithms are processed in the order of the creation priority on the profile type.

Contract Types And The Financial Design

In this section, we provide an example of how our contract types map to Bill Segment Types, Payment Segment Types, and Adjustment Profiles. This example is meant to help solidify the power of the financial model, it is not necessarily indicative of how your specific implementation will look.

Caution: If you are not comfortable with the topics described in *Defining Financial Transaction Options*, the following table will not make sense.

Division/ Contract Type	Distribution Code	Bill Segment Type	Payment Segment Type	Adjustment Profiles
CA/FS	A/R-BK	FS-RATED	NORMAL	FS
CA/Auto-PC	A/R-BK	FS-RATED	NORMAL	FS
CA/BK	A/R-BK	FS-RATED	NORMAL	FS

Setting Up Contract Types

In the previous section, Designing Contract Types, we presented a case study that illustrated a mythical organization's contract types. In this section, we explain how to use the windows on the Contract Type window group to maintain your contract types.

Contract Type - Main Information

Open **Admin Menu, Contract Type** and navigate to the **Main** tab to define core information about your contract types.

Caution: Every contract type is owned by a Division. This Division controls many values that can be referenced on the contract type. If you don't understand Divisions and their place in the application, do NOT attempt to set up your contract types. Rather, refer to *Setting Up Divisions* before proceeding.

Description of Page

Enter a unique combination of **Division** and **Contract Type** for every contract type.

Enter a **Description** for the contract type.

Service Type defines the type of service associated with the contract type. If the contract type has rates, only rates belonging to this service type may be linked to the contract type.

Fastpath: For more information about service types, refer to *Setting Up Service Types*.

Select the **Distribution Code** and **GL Division** that defines the receivable account for receivable-oriented contracts. For non-receivable oriented contracts, this distribution code is typically as follows:

- Deposits. The distribution code is a deposit payable account.
- Non-billed budgets. The distribution code is an overpayment payable account.
- Company usage. The distribution code is a company usage expense account.
- Write off. The distribution code is a write-off expense account.
- Payment arrangements. The distribution code is a payment arrangement receivable account.

Fastpath: For more information about GL accounts, refer to *The Source Of GL Accounts On Financial Transactions*.

Select the **Revenue Class** associated with the contract type (and its contracts). The revenue class may affect the revenue account(s) generated by the contract's rate.

Fastpath: Refer to *Rate Component - GL Distribution* for more information about revenue class.

Select the **Pay Segment Type** that defines how payment segments linked to contracts of this type affect:

- The contract's payoff and current balances

Fastpath: For more information about payment segment types, refer to *Setting Up Payment Segment Types*.

When a tender is canceled, a cancellation reason must be supplied. If the cancellation reason indicates a NSF (non sufficient funds) charge should be levied, the system invokes the Levy an NSF Charge algorithm specified on the tender's account's *customer class*. Because adjustments must be linked to a contract, the algorithm must determine the appropriate contract to use to levy the adjustment based on business rules. The charge is levied using the **NSF Adjustment Type** of the appropriate contract's contract type.

Caution: You must specify adjustment type profiles on the contract type (on the Adjustment Type window) before adjustment types will appear in the above drop downs.

Fastpath: For more information about adjustment types, refer to *Setting Up Adjustment Types*. For more information about cancellation reasons, refer to *Setting Up Payment Cancellation Reasons*.

Select the **Payment Priority**. This field is available for use by the algorithms that distribute partial payments amongst an account's contracts. Higher priority contracts will have their debt relieved before lower priorities. Refer to *Distribution*

Based on Payment Priority and *Delinquent Payment Distribution Algorithm* for information about payment distribution algorithms that use this field.

Note: The values for this field are customizable using the Lookup table. This field name is PAY_PRIORITY_FLG.

Fastpath: For more information about distribution priority, refer to *Distributing A Payment Amongst An Account's Contracts* .

Select the **Delinquent Pay** ment **Pr** iori **ty**. This field is available for use by the algorithms that distribute partial payments amongst an account's contracts. Higher priority contracts will have their debt relieved before lower priorities. Refer to *Delinquent Payment Distribution Algorithm* for information about a payment distribution algorithm that uses this field.

Note: The values for this field are customizable using the Lookup table. This field name is DEL_PRIORITY_FLG.

Turn on **Do Not Overpay** if the system is not allowed to distribute an overpayment to this type of contract (i.e., the contract is not allowed to have a system-created credit balance). This field is available for use by algorithms that distribute overpayments. Refer to *Overpayments Held On Highest Priority Contract* for information about an overpayment algorithm that uses this field.

Turn on **Late Payment Charge** if the system should generate a late payment charge for this type of contract if payment is not received on time. If this is turned on, you must define the **LPC Calc. Algorithm** used to calculate the late payment charge amount. Refer to *Defining Late Payment Charge Options* for more information about late payment charges. Examples of algorithm types used for calculating late payment charges are *BILPC-SPRC* and *BILPC-TOTAL*.

Define the **Adj** ustment **Type (Synch Curr)** that will be used to synchronize (make equal) the current amount with the payoff amount on a contract of this type. This type of processing happens as follows:

- Most *write-off* algorithms that perform financial efforts (e.g., writing off debt), will issue an adjustment of this type if the contract's current and payoff balances are not equal.
- If a user stops a customer on a budget plan, the system issues adjustments of this type to synchronize the customer's current and payoff balances.
- If a user stops a contract covered by a *non-billed budget*, the system issues adjustments of this type to synchronize the customer's current and payoff balances.
- If a *cancellation of a bill segment* occurs after a customer has stopped participating in a budget plan, an adjustment of this type is issued to synchronize the imbalance created when the bill segment's financial transaction is canceled.

Turn on **CIAC Contract Type** and specify an appropriate **CIAC Refund Process** if contracts of this type are used to bill for Contribution In Aid of Construction (CIAC) charges.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SA_TYPE*.

Contract Type - Detail

Open **Admin Menu, Contract Type** and navigate to the **Detail** tab to define additional details about a given contract type.

Description of Page

Turn on **Display As Alert** if Control Central should display an alert if an account has a contract of this type that isn't `Closed` or `Canceled`. If this switch is on, also enter the **Alert Information** to appear on Control Central. We recommend only using this feature on unusual contract types (e.g., payment arrangements, write-offs) so that a CSR is not presented with an alert for every contract type.

If this contract type is used for any of the **Special Role** s defined in the drop down, indicate which one. Valid values are: `Billable Charge`, `Cash Deposit`, `Interval`, `Loan`, `Non-billed Budget`, `Payment Arrangement`, `Write Off`. This information is used on windows with functionality that can only be used by contracts used for specific roles. For example, the Billable Charge window group can only reference `Billable Charge` contracts.

If Special Role is `Cash Deposit`, you must define the **Deposit Class** of the deposit. You should also define a **Deposit Class** on every contract type to which a given deposit can be distributed.

Fastpath: Refer to *What Do Deposit Classes Do?* for more information.

If the Special Role is `Loan`, you must also define the following fields:

- Use the **Interest Bill Factor** to define the bill factor code for the loan interest rate.
- Use **Override Interest Flag** to indicate whether the interest rate defined on the interest bill factor may be overridden at the contract level. If you select `Allowed`, the interest rate may be overridden by a contract value on the contract.
- Use the **Loan A/R Distribution Code** to define the distribution code to be used when posting the short-term receivable amount to the general ledger (the normal distribution code is used for the long-term receivable). If the normal distribution code is the same as the **Loan A/R Distribution Code**, the contract type does not differentiate between long- and short-term receivables. If the two distribution codes are different, the contract type differentiates between long- and short-term receivables.

Fastpath: Refer to *Defining Loan Options* for more information about **Interest Bill Factor**, **Override Interest Flag** and **Loan A/R Distribution Code**.

If the Special Role is `Non-billed Budget`, you must also define the following:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to the transfer accumulated credit from the non-billed budget contract to the contracts covered by the non-billed budget when the account is billed or the non-billed budget contract is stopped.
- Use the **Non-billed Budget Monitoring** to specify whether the non-billed budget is monitored by the account debt monitor.

If the contract type is defined as `Eligible for Non-billed Budget`, you must also define the following:

- Use **Adjustment Type (Current=0)** to specify the type of adjustment used to set the contract's current balance to zero when a contract of this type is linked to an active, monitored non-billed budget.

If the Special Role is `Payment Arrangement`, you must also define two adjustment types:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to transfer funds from the delinquent contracts to the payment arrangement contract.
- Use **Adjustment Type (Current=0)** to specify the type of adjustment used to set the payment arrangement's current balance to zero after funds have been transferred.

If the Special Role is `Write Off`, you must also define the following adjustment types:

- Use **Adjustment Type (Xfer)** to specify the type of adjustment used to transfer funds from the uncollectable contracts to the write off contract.

Caution: You must specify adjustment type profiles on the contract type (on the Adjustment Type window) before adjustment types will appear in the above drop downs.

Turn on **One Time Charge** if this contract type is used for one-time invoices. When a one-time invoice contract is created, the system sets the stop date of the contract to be equal to the start date.

Renewal of contracts of this type may be `Optional`, `Not Allowed` or `Required` depending on your business processes. If renewal is not allowed, the contract expires on the expiration date. Renewal treatment is an important consideration for contracts that require an expiration date.

If renewal is required or optional, specify the **Days Before Expiration for Renewal**. Note that currently this is only used by non-billed budgets to calculate the renewal date based on the expiration date.

If the Special Role is `Non-billed Budget`, **Non-billed Budget Monitoring** must indicate whether the non-billed budget is monitored by the account debt monitor.

Where Used

The alert information is used by Control Central to alert a CSR when unusual contracts exist for an account.

Only contract types designated as being `Billable Charge` may have billable charges linked to them. Refer to *Maintaining Billable Charges* for more information.

Only contract types designated as being `Cash Deposit` are processed by the various deposit-related background processes (e.g., interest calculation, automatic refund, etc.).

Only contract types designated as `Loan` are used to define the loan terms for a loan contract. Refer to *Loans* for more information.

Only contract types designated as `Non-billed Budget` may be used to set up non-billed budgets. Refer to *Non-billed Budgets* for more information.

Only contract types designated as being `Payment Arrangement` may be used on the payment arrangement window group. Refer to *Setting Up Payment Arrangements* for more information.

Only contract types designated as being `Write Off` may be specified as the write off contract type on distribution codes. Refer to *Setting Up Distribution Codes* for more information.

Only contracts whose contract type is designated as being `Write Off` appear on the Write Off contract's query. Refer to *Write Off - Write Off Contracts* for more information.

Contract Type - Billing

Open **Admin Menu, Contract Type** and navigate to the **Billing** tab to define how the system manages bill segments for contracts of a given contract type.

Description of Page

Turn on **Eligible for Billing** if the system should create bill segments for contracts of this type. This will typically be turned on for all contracts except for those used to hold write-off amounts or to levy one-off adjustments.

Define the minimum number of days a bill segment (other than the final segment) must span using **Minimum Days for Billing**. This is useful to prevent initial bill segments that span only a few days.

Fastpath: For more information about minimum days, refer to *Preventing Short Bill Segments*.

Select the **Bill Segment Type** that controls both how bill segments for this contract type will be created and how the related financial transaction affects the general ledger and the customer's debt.

Fastpath: For more information about bill segment types, refer to *Setting Up Bill Segment Types*.

Use **Default Description on Bill** to define the verbiage that should print on the customer's bill.

Note: Rates overwrite this description. The Default Description on Bill is not applicable for contracts whose charges are calculated using a rate. Why? Because the description that appears on the bill segment is defined on the rate schedule's rate version.

Note: Billable charges overwrite this description. The Default Description on Bill is not applicable for contracts whose charges are calculated using a billable charge. Why? Because the description that appears on the bill segment is defined on the billable charge.

Use the **Billing Processing Sequence** if you need to control the order in which contracts linked to this contract type are processed by billing and interval data creation processes.

Fastpath: Refer to *Processing Sequence Considerations* for more information.

Use **Bill Print Priority** to define the order in which the contract type's bill segments should appear on bills (relative to the other contract types that appear on a bill).

Note: The values for this field are customizable using the Lookup table. This field name is BILL_PRT_PRIO_FLG.

Use **Max Bill Threshold** if you want the system to generate a bill error when a bill segment is produced *in batch* that exceeds a given value. These bill errors will appear on the standard billing queries and To Do lists. If, after reviewing the high value bill segment, an operator truly intends to send the bill out, they should regenerate the bill. Refer to *How To Correct A Bill Segment That's In Error* for more information.

Caution: The value entered in this field will DEFAULT onto contracts of this type when they are first created. An operator may change the default value on a contract in case a specific customer has unusually high bills that continually error out. It's important to be aware that if you change the value of High Bill Amount on an contract type and there already exist contracts of this type, the existing contracts will contain the original value (the new value on the contract type will not be propagated on the existing contracts).

Use **Graph Unit Of Measure** to define the unit of measure of the graphed consumption on the bill (if any).

Turn on **Allow Estimates** if the system is allowed to generate estimated consumption at billing time. This value is defaulted onto contracts and can be overridden on an individual contract.

Turn on **Characteristic Location Required** if a characteristic location must be linked to the contract when the contract is activated. The characteristic location is used to define the taxing authorities associated with the contract's bill segments. It is also used to identify where the contract's service is located on various windows.

Fastpath: For more information about how characteristic location is used, refer to *An Illustration Of A Bill Factor And Its Characteristics*.

Use the **Initial Start Date Option** to control how billing should calculate the consumption period for the very first bill for contracts of this type. This field is not applicable for sub contract types or contract types with a special role of `Billable Charge`. Valid values are `Include First Day`, `Add 1 Day Always` and `Add 1 Day for Back-to-back`.

Contracts may have the end date of their bill segments defined on a user-maintained bill period schedule. This option is used when bill segments must fall on strict calendar boundaries (e.g., quarterly bills that end on the last day of the quarter). If this contract type should be billed like this, select `Use Bill Period` in the **Use Calendar Billing** field. When this option is used, you must define the **Bill Period** whose schedule defines the bill segment end dates.

Fastpath: For more information about bill period schedules, refer to *Designing Bill Periods*. For more information about other bill end date methods, refer to *Ways To Control The End Date Of A Bill Segment*.

Instead of the `Use Bill Period` method, contracts may have their bill segment end date based on the first day of service. For example, if service started on the 16 th of some month, the ongoing bill segments will start on roughly the 16 th of each month. If this contract type should be billed like this, select `Anniversary Future Billing` or `Anniversary Past Billing` in the **Use Calendar Billing** field. When either option is used, you must define the **Anniversary Bill Frequency**. This frequency defines the amount of time between bill segments.

Fastpath: For more information about anniversary billing, refer to *Using The Anniversary Method*. For more information about other bill end date methods, refer to *Ways To Control The End Date Of A Bill Segment*.

If you want to create billable charges with future date for a contract, you must select the **Allow Advanced Billing** option from the **Use Calendar Billing** list.

Total Bill Amount indicates whether contracts of this type can use the total amount to bill field on the contract page. Valid values are `Not Allowed` and `Required`. Only contract types used to bill for deposits or loans should have this field set to `Required`.

If `Required` is selected, you must enter the **Total Amount To Bill Label**. The **Total Amount To Bill Label** defines the label that prefixes the total bill amount on the contract page for contracts of this contract type.

Recurring Charge indicates whether contracts of this type can use the recurring charge field on the contract window . Valid values are `Not Allowed`, `Optional` and `Required`. If either `Optional` or `Required` are used, you must enter:

- **Recurring Chg Amt Label**. This defines the label that prefixes the recurring charge amount on the contract window for contracts of this contract type.
- **Recurring Charge Frequency**. This defines the following:
  - Specifies the frequency at which the Recurring Charge Amount specified on contracts of the contract type is to be billed.
  - Serves as the basis for proration of the Recurring Charge Amount.
  - Specifies the frequency at which contracts of the contract type without a rate will be billed.

Fastpath: For more information about how to use the recurring charge information, refer to *Merchandise Segmentation - Installment Billing*, *Deposit Segmentation - Installment Billing*, *Payment Arrangement Segmentation*, *The Terms Of A Loan Are Stored On A Contract*.

Turn on **Eligible for Budget** if contracts of this type can participate in budget billing. If this switch is turned on, then you must define the **Adj** ustment **Type (Synch Current)** that will be used to synchronize (make equal) the current amount with the payoff amount on a contract of this type when a budget is cancelled. (The Adjustment Type (Synch Current) field is on the main page.)

Fastpath: Refer to *Budget Billing* for more information about budgets in general. Refer to *Designing and Defining Budget Plans* for more information.

Set the **Eligible for Non-billed Budget** flag to `Eligible for Non-billed Budget` if you want contracts of this type to be eligible to be covered by a non-billed budget. If this flag is set to `Eligible for Non-billed Budget`, you must also define the **Adjustment Type (Current = 0)** field (on *Contract Type - Detail*).

Require Total Amount Switch versus Bill Segment Algorithm

The following table shows valid combinations of the contract type's required total amount switch and the bill segment creation algorithm defined on the contract type's bill segment type. If N/A appears in a cell, the combination is not supported in the system. Otherwise, we list typical types of contracts that will use a combination.

Contract Type Require Total Amount Switch **Bill Segment Create Algorithm**	**Not allowed**	**Required**
Apply Rate	Financial services. Other Financial services. Misc recurring charges whose value is specified in a rate or is taxable.	N/A
Recurring Charge With Auto Stop	Payment arrangements. Merchandise installment plans.	N/A
Recurring Charge For Amount To Bill	N/A	Deposit installment plans.
Billable Charge	One time invoices. Pass through charges	N/A
Loan	N/A	Loans.

Allow Recurring Charge Switch versus Bill Segment Algorithm

The following table shows valid combinations of the contract type's allow recurring charge switch and the bill segment creation algorithm defined on the contract type's bill segment type. If N/A appears in a cell, the combination is not supported in the system. Otherwise, we list typical types of contracts that will use a combination.

Contract Type Recurring Charge Switch **Bill Segment Create Algorithm**	**Not allowed**	**Optional**	**Required**
Apply Rate	Financial services - no budget Misc recurring charges whose value is specified in a rate or is taxable.	Financial services - budget optional	Financial services - budget required

Contract Type Recurring Charge Switch **Bill Segment Create Algorithm**	**Not allowed**	**Optional**	**Required**
Recurring Charge With Auto Stop	N/A	Payment arrangements. Merchandise installment plans. SEE NOTE!	Payment arrangements. Merchandise installment plans. SEE NOTE!
Recurring Charge For Amount To Bill	N/A	Deposit installment plans. SEE NOTE!	Deposit installment plans. SEE NOTE!
Billable Charge	One time invoices. Pass through charges	N/A	N/A
Loan	N/A	N/A	Loans.

Note: Most recurring charge contract types require a recurring charge amount on their contracts. However, the above matrix indicates you can have recurring charge contract types where this value is optional. Why? A special algorithm exists in billing that says if the recurring charge amount is 0 (zero) the system will bill the remaining payoff balance or total amount to bill. This algorithm exists so that you can easily bill the amount in one lump sum (i.e., don't bill it in installments).

Where Used

The billing information is used when the system creates a bill segment for contracts of this type.

Contract Type - Rate

Open **Admin Menu, Contract Type** and navigate to the **Rate** tab to define the rates that may be referenced on contracts of a given type.

Description of Page

Turn on **Rate Required** if the bill segment creation algorithm for the contract type expects a rate schedule to be referenced on contracts of this type.

Fastpath: For more information, refer to *Rates*.

Define the date the system uses when selecting an effective-dated rate (from the contract's rate history) using **Rate Selection Date**. Selecting `Bill Start Date` will cause the system to use the rate effective on the first day of the bill segment's period. Selecting `Bill End Date` will cause the system to use the rate effective on the last day of the bill period. Selecting `Accounting Date` will cause the system to use the rate effective on the accounting date associated with the bill.

The information in the **Rate Schedules** collection defines the rates that may be referenced on contracts of this type. The following fields are required for each contract type:

Rate Schedule Specify the primary rate schedule; its description is displayed adjacent.

Use As Default Turn on this switch for the rate to be defaulted on new contracts.

Rate Required versus Bill Segment Algorithm

The following table shows appropriate combinations of the contract type's rate required switch and the bill segment creation algorithm defined on the contract type's bill segment type. If N/A appears in a cell, the combination is not applicable. Otherwise, we list typical types of contracts that will use a combination. Be aware that no cross validation exists between the rate required switch and the bill segment creation algorithm when setting up the contract type.

Contract Type Rate Required Switch **Bill Segment Create Algorithm**	**Not allowed**	**Rate required on contract**
Apply Rate	N/A	Misc item services. Company usage.
Recurring Charge With Auto Stop	Payment arrangements. Merchandise installment plans. Zero-interest loans.	N/A
Recurring Charge For Amount To Bill	Deposit installment plans.	N/A
Recurring Charge	Charitable contributions.	N/A
Billable Charge	One time invoices.	Billable charges that require a rate to add-on extra charges (like taxes) or billable charges where the consumption is interfaced and the system is responsible for calculating the charges.
Loan	Loans.	N/A

Where Used

This information is used to default and validate the rate specified on a contract. Refer to *Contract - Rate Info* for more information.

Contract Type - Adjustment Profiles

Open **Admin Menu, Contract Type** and navigate to the **Adj Profile** tab to define the adjustment profiles that define adjustment types that may be referenced on contracts of a given type.

Description of Page

Define the **Adjustment Type Profiles** that, in turn, define adjustment types that may be referenced on contracts of a given type.

Fastpath: For more information about adjustment type profiles, refer to *Setting Up Adjustment Type Profiles*.

Where Used

This information is used to validate the adjustments linked to the contract. Refer to *Adjustments - Main Information* for more information.

Contract Type - C&C

Open **Admin Menu, Contract Type** and navigate to the **C&C** tab to maintain attributes that affect how the system severs the contract when collection attempts fail.

Description of Page

Select the **Debt Class** associated with the contract type. Any debt on a contract of this contract type will be categorized under this debt class.

Select the **Write Off Debt Class Code** associated with the contract type. Any debt on a contract of this contract type will be categorized under this debt class during write-off processing.

Note: Write Off Debt Class vs. Regular Debt Class. It's important to be aware that a contract type references both a regular debt class and a write-off debt class. The regular debt class controls the collection criteria applied against an account's contracts. The regular debt class is also used to segregate an account's outstanding balance on several queries in the system. The write-off debt class controls the write-off criteria applied against an account's stopped contracts. The reason the system supports two different debt classes is because you may categorize your contracts differently when you try to collect overdue debt versus when you write-off debt. Refer to *The Big Picture Of Write Off Processing* for more information.

Where Used

The debt class has multiple uses:

- The system summarizes an account's debt by debt class on *Account - Main Information* and *Account - Financial Balances*.
- Debt class is one component that controls how the system analyzes an account's overdue debt (the others are the account's collection class and currency). Refer to *Different Collection Criteria For Different Customers And Different Debt* for more information.
- Write off debt class is one component that control how the system writes off an account's stopped contracts. Refer to *Different Write-Off Criteria For Different Customers And Different Debt* for more information.

Contract Type - Billable Charge Template

Open **Admin Menu, Contract Type** and navigate to the **BC Template** tab to define the billable charge templates that can be used on contracts of a given type.

Note: Only billable charges have billable charge templates. Only contracts that are defined as Billable Charges (in the Special Role on the Details window) may use the grid on this window.

Description of Page

The information in the **Billable Charge Template** collection defines the contract type's permissible billable charge templates. A billable charge template contains the default bill lines, amounts and distribution codes used to levy a one-off charge. The following fields are required for each template:

Billable Charge Template Specify the billable charge template. Its description is displayed adjacent.

Use As Default Turn on this switch for the template to be defaulted on new billable charges linked to contracts of this type (if any).

Fastpath: For more information about billable charge templates, refer to *Setting Up Billable Charge Templates*.

Where Used

This information is used to limit the billable charge templates that can be used for a given contract type.

Contract Type - Characteristics

To define characteristics common to all contracts of a given type, open **Admin Menu, Contract Type** and navigate to the **Characteristics** tab.

Description of Page

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all contracts of this type.

Note: You can only choose characteristic types defined as permissible on a contract type record. Refer to *Setting Up Characteristic Types & Their Values* for more information.

Contract Type - Algorithms

Open **Admin Menu, Contract Type** and navigate to the **Algorithm** tab to define the algorithms that should be executed for contracts of a given type.

Description of Page

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

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

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

The following table describes each **System Event** for which you can define algorithms.

System Event	Optional / Required	Description
`Bill Completion`	Optional	These algorithms are executed whenever a bill is completed for an account that contains a non-canceled contract of this type. The following situations necessitate the definition of a completion algorithm on an contract type: - As explained under *Overpayment Segmentation*, when a bill is completed, the system may apply an excess credit from a prior overpayment to an account's contracts. **Note**. Algorithms of this type are called for all non- `Canceled` contracts, regardless of whether or not they are billed. If your algorithms should only be processed under certain conditions (for example, only process this algorithm for `Active` contracts), then it is the responsibility of the algorithm to check the conditions before continuing. Click *here* to see the algorithm types available for this system event.
`Budget Eligibility`	Optional	These algorithms are executed when determining in a contract is eligible for budget. Algorithms of this type are only applicable on contract types that are marked as eligible for budget and may be used to override that setting and indicate that the contract is not eligible. For example, maybe contracts in a certain rate are not eligible. Or perhaps contracts with a given characteristic value are not eligible. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`FT Freeze`	Optional	These algorithms are executed whenever a financial transaction is frozen that is linked to a contract of this type. Click *here* to see the algorithm types available for this system event.
`Loan Interest Charge`	Optional	These algorithms are executed whenever the interest charge needs to be calculated for a loan, such as when the loan amortization schedule is created and when a bill segment is created for a loan contract. This algorithm should be specified on contract types with a special role of `Loan`. Refer to *Defining Loan Options* for more information. Click *here* to see the algorithm types available for this system event.
`Loan Periods and Amount`	Optional	These algorithms are executed whenever a user clicks the Calculate button on *Loan - Main* or on the Start Contract Confirmation dialog for a loan contract. It calculates either the number of periodic payments or the payment amount (depending on whether the user specifies the number of payments or the payment amount as input). This algorithm should be specified on contract types with a special role of `Loan`. Refer to *Defining Loan Options* for more information. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Loan Schedule`	Optional	Thesealgorithms are executed whenever the system needs to create a *loan amortization schedule* for a loan, such as when you renegotiate the terms of a loan on *Loan - Main*. This algorithm should be specified on contract types with a special role of `Loan`. Refer to *Defining Loan Options* for more information. Click *here* to see the algorithm types available for this system event.
`Payment Arrangement`	Optional	These algorithms are by executed to handle the creation, breaking and canceling of payment arrangement contracts. This algorithm should be specified on contract types with a special role of Payment Arrangement to perform special actions that take place during the lifecycle of a payment arrangement. Refer to Monitoring Payment Arrangements for more information about payment arrangements. Note that the Payment Arrangement algorithm and the Break Pay Arrangement algorithm are mutually exclusive.
`Payment Freeze`	Optional	These algorithms are executed whenever a payment is frozen. The following situations necessitate the definition of such an algorithm: - For a loan contract, such an algorithm is required to create a frozen adjustment that transfers any credit balance resulting from an *overpayment* to the loan's principal balance. Refer to *Defining Loan Options* for more information. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Pre-Bill Completion`	Optional	These algorithms are executed immediately prior to bill completion when a bill contains a bill segment for a contract whose contract type has such an algorithm. The following situations necessitate the definition of such an algorithm: - If you want to delete a bill segment that's in error on the last night of a bill cycle when there are other bill segments that aren't in error, use such an algorithm. Click *here* to see the algorithm types available for this system event.
`Process NBB Scheduled Payment`	Optional	These algorithms are executed by the NBB Scheduled Payment Processing background process whenever a scheduled payment is due. If the non-billed budget contract is unmonitored, this algorithm is not called. This algorithm should be specified on non-billed budget contract types to create the necessary adjustments for the non-billed budget contract. Click *here* to see the algorithm types available for this system event.
`Proposal Contract Acceptance`	Optional	These algorithms are executed when a *proposal contract* is accepted. Refer to *Enabling The Creation Of A Real Contract* for more information. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Proposal Contract Bill Segment Generation`	Optional	These algorithms are executed to generate simulated bills segments for a *proposal contract*. Refer to *Enabling The Generation Of Simulated Bill Segments* for more information. Click *here* to see the algorithm types available for this system event.
`Proposal Contract Creation`	Optional	These algorithms are executed when a *proposal contract* is created. Refer to *Enabling The Automatic Generation Of Billing Scenarios* for more information. Click *here* to see the algorithm types available for this system event.
`Contract Activation`	Optional	These algorithms are executed when a contract status changes from `Pending Start` to `Active`. It performs any additional activities that are necessary to activate an contract. The following situations necessitate the definition of such an algorithm: - If you want to create a customer contact to indicate that a non-billed budget has been activated, use such an algorithm. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Contract Cancel`	Optional	These algorithms are executed when a contract status changes to `Canceled`. It performs any additional activities that are necessary to cancel an contract. An example of when you may use this algorithm is that perhaps your business rules dictate that the creation of a payment arrangement should create a credit rating history transaction. When a payment arrangement contract is canceled, the credit rating should be updated with an end date. Click *here* to see the algorithm types available for this system event.
`Contract Creation`	Optional	These algorithms are executed when a contract is created. The following situations necessitate the definition of such an algorithm on an contract type: - If you want to create a To Do entry whenever a new contract of a given type is added, specify such an algorithm. - If you want to automatically activate contracts of a given type (instead of waiting for the background contract activation process to run), specify such an algorithm. - If you want to create a Workflow Process when a contract of a given type is added, specify such an algorithm. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Contract Information`	Optional	We use the term "contract information" to describe the basic information that appears throughout the system to describe a contract. The data that appears in "contract information" is constructed using this algorithm. Plug an algorithm into this spot to override the "contract information" algorithm on installation options or the system default "contract information" if no such algorithm is defined on installation options. Click *here* to see the algorithm types available for this system event.
`Contract Renewal`	Optional	These algorithms are executed by the contract Renewal background process whenever an contract is due for renewal or when the user clicks the **Renew** button (for *non-billed budgets*). It performs any activities that are necessary to renew an contract and returns the new renewal and expiration dates for the contract. Click *here* to see the algorithm types available for this system event.
`Contract Stop`	Optional	These algorithms are executed whenever a contract's status changes from `pending stop` to `stopped`. The following situations necessitate the definition of a contract stop algorithm: - For service credit memberships that have a *refundable membership fee*, an Contract Stop algorithm attempts to refund the fee if this is the last contract linked to the membership that is being stopped. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Contract Stop Initiation`	Optional	These algorithms are executed whenever a contract's status becomes `pending stop`. The following situations necessitate the definition of a stop initiation algorithm on an contract type: - As explained under *Finalizing Pending Stops*, contracts are normally transitioned from `pending stop` to `stopped` by a background process (or manually). **When does a contract become pending stop?** contracts typically become `pending stop` when a user initiates a request to stop service on *Start Stop - Main*. Click *here* to see the algorithm types available for this system event.

Contract Type - Billable Charge Overrides

The *BCU2 - Create Billable Charge* background process is responsible for creating billable charges for each billable charge upload staging record interfaced into the system. This process will override the values of the various switches referenced on a bill charge upload staging line if the respective contract's contract type has an override value for the bill charge upload staging line's billable charge line type.

Note: This information is optional. If you don't need to override the values of a *Billable Charge Line Type* you don't need to set up this information.

Open **Admin Menu, Contract Type** and navigate to the **BC Upload Override** tab to define override values for a given Contract Type / Billable Charge Upload Staging Line Type.

Description of Page

Use the **Billable Charge Overrides** collection to define values to be overridden on billable charge lines uploaded from an external system (refer to the description above for the details). The following switches may be overridden for a given **Contract Type** and **Billable Charge Line Type.**

- Use the **Show on Bill** switch to define the value to be defaulted into the Show on Bill indicator on billable charge upload lines that reference this line type.
- Use the **Appear in Summary** switch to define the value to be defaulted into the App in Summary indicator on billable charge upload lines that reference this line type.
- Use **Memo Only, No GL** switch to define the value to be defaulted into the Memo Only, No GL indicator on billable charge upload lines that reference this line type.
- Use **Distribution Code** to define the value to be defaulted into the Distribution Code on billable charge upload lines that reference this line type.

Contract Type - Interval Info

Open **Admin Menu, Contract Type** and navigate to the **Interval Info** tab to define the interval profile relationship types and Variance Parameter map relationship types, which are valid for contracts of a given type.

Note: This tab may not appear. This tab is suppressed if the interval billing `Complex Billing` module is *turned off*.

Description of Page

If the contract type's special role is `Interval`, you may define the **I nterval Profile Relationship Types** that may be linked to contracts of this type.

If the contract type's special role is `Interval`, you may define the **Variance Parameter Map Relationship Types** that may be linked to contracts of this type.

Where Used

The interval profile information is used to validate the interval profile relationship types linked to the contract. And, the Variance Parameter map information is used to validate Variance Parameter map relationship types linked to a contract.

Contract Type - NBB Recommendation Rule

Open **Admin Menu, Contract Type** and navigate to the **NBB Rec'n Rule** tab to define the recommendation rules that are valid for non-billed budget contracts of this type.

Description of Page

If the contract type's special role is `Non-billed Budget`, you may define the **Recommendation Rules** that are valid on non-billed budget contracts of this type. Check the **Use As Default** box to indicate the default recommendation rule for contracts of this type.

Fastpath: For more information about non-billed budgets, refer to *Defining Non-Billed Budget Options*.

Where Used

The non-billed budget recommendation rules are used to recommend the payment amount and payment schedule for non-billed budget contracts. Refer to *Maintaining Non-billed Budgets* for more information.

Contract Type – Contract Category

The **Contract Category** tab allows you to map a product, risk, or a coverage (defined in the external system) to the contract type defined in Oracle Revenue Management and Billing. This tab contains the following fields:

Field Name	Field Description
Source	Used to specify the name of the external system from where the product originated.
Product Category	Used to indicate the category of the product. The valid values are: • Product • Risk • Coverage
Product Type	Used to specify the type of product.
Product Code	Used to specify the product code.
Coverage Type	Used to specify the type of coverage.
Sub-LOB	Used to specify the line of business where the product is used.

Chapter 8

Background Processes Addendum

Topics:

This chapter is an addendum to the **Defining Background Processes** chapter in the **Oracle Utilities Application Framework Administrative Processes** section. This addendum describes the background processes that are provided with Oracle Revenue Management and Billing.

The System Background Processes

The topics in this section describe functionality that is common to system background processes.

Process What's Ready Processes

Some background processes create and update records that are "ready for processing". The definition of "ready" differs for every process. For example,

- The bill cycle process produces bills for all accounts belonging to open bill cycles.
- The account debt monitor process analyzes the debt associated with all accounts whose review date is on or before the business date.
- The process that activates pending stop and pending start contract's attempts to activate all contracts that aren't already activated.

Processes of this type tend to use a business date in their determination of what's ready. For example, the bill cycle process creates bills for all bill cycles whose bill window is open (i.e., where the business date is between the bill cycle's start and end date). If the requester of the process does not supply a specific business date, the system assumes that the current system date should be used. If you need to use a date other than the current date, simply supply the desired date when you request the batch process.

The following table lists every background process that processes all data that is "ready".

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
ACTVTAPY	CIPPAAPB	This process marks each auto pay download staging record with the batch control associated with its auto-pay source's route type. It also stamps the respective batch control's current run number on each record. Note: The APAYACH/C1-APACH background processes use the information on this staging table to create the flat file that is used to interface information to the ACH. The BALAPY background process uses the information on this staging table to create automatic payment tender controls. Refer to *Activating Automatic Payments* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
ADM	CIPLADMB	The account debt monitor analyzes all accounts whose C&C review date is on or before the supplied business date. Refer to *The C&C Monitors* for more information. The input parameter controls how the trigger date is set on collection events that are created by this process. Refer to *Calendar vs Work Days* for more information about your date arithmetic options.	Yes	ADD-WORK-DAYS (Y or N) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
ADM2	CIPLDM2B	The account debt monitor analyzes all accounts who have not been analyzed in the last X days (where X is the Days Between Review defined on the account's customer class). Refer to *The C&C Monitors* for more information. The input parameter controls how the trigger date is set on collection events that are created by this process. Refer to *Calendar vs Work Days* for more information about your date arithmetic options.	Yes	ADD-WORK-DAYS (Y or N) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
APAYCRET	CIPPACRB	This process creates automatic payments for bills whose automatic payment creation has been deferred until the extract date. This extract date is stamped on the bill and is used by this background process to select all bills whose automatic payment extract date is on or before the supplied business date. It calls the automatic payment creation algorithm plugged in on the installation record to create the automatic payments. Note that the algorithm supplied does not distribute and freeze the automatic payments that are created. This is handled by the complementary background process APAYDSFR. Refer to *Installation Options - Billing* and *Automatic Payments* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
APAYDSFR	CIPPADFB	This process distributes and freezes automatic payments whose distribution date (indicated on the download staging record) is on or before the supplied business date. Payments that have been distributed (e.g., manually) are frozen if the above criterion is satisfied. This job complements the `APAYCRET` background process and the `PPAPAY` background process when the **Autopay Creation Option** on the installation record is set to `Create on Extract Date`. Refer to *Installation Options - Billing* and *Automatic Payments* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
ASSGNSBN	CIPBASBB	This process allocates completed bills a sequential bill number. You need only schedule this job if your organization assigns sequential bill numbers. Please refer to *Sequential Bill Numbers* for important information about this job and why it may not be necessary if you single-thread the `BILLING` background process.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	1/ not applicable

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
BALAPY	CIPPBAPB	This process creates a new tender control (with an associated deposit control) for each batch control and run number encountered for extracted automatic payments that are not already linked to a tender control. Afterwards, this process balances the open tender and deposit control records. Note: Automatic payment staging records are activated by the ACTVTAPY process and extracted by either APAYACH or C1-APACH. Refer to *Creating Automatic Payment Tender Controls* for more information.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
BCASSIGN	CIPFBCAB	This process assigns the `Pending` balance control group to new FT's (i.e., those without a balance control group). Refer to *The Big Picture of Balance Control* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
BCGNEW	CIPFBCGB	This process creates a `Pending` balance control group if one doesn't already exist. Refer to *The Big Picture of Balance Control* for more information.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	N/A (only 1 record is inserted)
BCGSNAP	CIPFBCSB	The balance control snapshot and verification process has two functions: 1. It summarizes the number and value of the financial transactions on the current `Pending` balance control group record. 2. It verifies the financial integrity of your system.	No	VERIFY-ONLY-SW MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		The value of the VERIFY-ONLY-SW parameter controls which of these functions is performed: - If VERIFY-ONLY-SW = "N", the system summarizes the new financial transactions under the current `Pending` balance control and verifies that the balances summarized on *every* historical balance control group are consistent with the financial transactions associated with this balance control group (i.e., it checks the financial integrity of the system). - If VERIFY-ONLY-SW = "G", the system only summarizes the new financial transactions under the current `Pending` balance control (i.e., the verification step is not performed).				

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		- If VERIFY-ONLY-SW = "Y", the system verifies that the balances summarized on *every* historical balance control group are consistent with the financial transactions associated with this balance control group (i.e., it checks the financial integrity of the system). Note: You may want to use the VERIFY-ONLY-SW parameter to improve system performance. For example, you can generate the balance control summary nightly (run the process with the switch set to "G") and validate the balance control summaries weekly (run the process with the switch set to "Y"). Refer to *The Big Picture of Balance Control* for more information.				

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
BCU1	CIPCBC1B	The first phase of the billable charge upload staging process validates and defaults information on to billable charge upload staging records. Refer to *Billable Charge Upload Background Processes* for more information.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	N/A
BCU2	CIPCBC2B	The second phase of the billable charge upload staging process creates billable charges for the new billable charge upload staging records. Refer to *Billable Charge Upload Background Processes* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15
BILLING	CIPBBILB	The bill cycle process creates bills for accounts with an "open" bill cycle. Refer to *Batch Billing* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
BUDMON	CIPGMBGB	The budget monitor analyzes all customers with a budget plan and highlights those where the current budget amount is out-of-sync with the recommended budget amount. Refer to *Budget Billing* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
BUDTRUP	CIPGTUPB	The budget true up process periodically trues up customers on a budget plan. Refer to *Budget Billing* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
C1-ADMOV	CIPLOVMB	The overdue monitor uses your overdue rules to collect overdue debt. Refer to *How Does The Overdue Monitor Work?* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	2000/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
C1-CSTRS	CIPQTRCB	The case scheduled transition process transitions cases to a nominated **next status** or **transition condition** at a scheduled time. The process selects all open cases whose current status is linked to the process' batch control code and are allowed to transition from their current status to the chosen next status or condition (i.e. where a corresponding transition rule exists for the case type/status combination) based on the input algorithm parameters.	Yes	NEXT-STATUS-CD (Next Status Code) NEXT-TR-COND-FLG (Next Transition Condition) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
C1-ODET	CIPLOETB	The overdue event manager activates all overdue events whose trigger date is on or before the supplied business date. Refer to *How and When Events Are Activated* for more information. This process also has the responsibility of recursively activating later events that are dependent on the completion of earlier events. For overdue events that are in the `Wait` state, this process runs the associated waiting algorithm for the event type to determine if the object the event is waiting for is complete (and then triggering the dependent events when it completes). Populate an Overdue Process Template in the input parameter to limit the processing to overdue processes for this template.	Yes	OD-PROC-TMP-CD (Optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	2000/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
C1-PEPL1	CIPPEL1B	This is the first of three background processes that load the contents of the payment event upload staging records into the various payment tables. It first creates new deposit and tender control records, and then updates the payment event upload staging records with the corresponding Tender Control ID. Next, it processes each `incomplete` record as follows: It updates the record's Tender Account ID with the account ID returned by the `Determine Tender Account` algorithm defined on the *distribution rule* . If the Pay Event Process ID field is not populated, it is set equal to the tender account ID. If no error was encountered, it transitions the record from to `Pending`.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		Refer to *Interfacing Payments Using Distribution Rules* for more information.				
C1-PEPL2	CIPPEL2B	This is the second of three background processes that load the contents of the payment event upload staging records into the various payment tables. The responsibility of this process is to create payment events, payment tenders and payments and transition the corresponding staging records from `Pending` to `Complete`. Refer to *Interfacing Payments Using Distribution Rules* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
C1-PEPL3	CIPPEL3B	This is the last of three background processes that load the contents of the payment event upload staging records into the various payment tables. The responsibility of this process is to update the status of the related deposit and tender controls from `open` to `balanced`. Refer to *Interfacing Payments Using Distribution Rules* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
C1-WFSUB	CIPWWETB	This process does two things: It sets the trigger date of workflow events for the input **workflow process template** that are dependent on the completion of earlier workflow events. It activates all workflow events for the input **workflow process template** whose trigger date is on or before the supplied business date. This background process is the same code used for WFET. It is used for the batch scheduling functionality. Refer to *Workflow Event Dependencies* for more information.	Yes	WF-PROC-TMPL-CD (Workflow Process Template) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
CAREPROG	CIPCCRCB	This batch process is responsible for creating customer contacts (letters) for contract characteristics that are about to expire. It finds contracts with the indicated Characteristic Type and Value that will expire within Threshold Days and creates the indicated type of customer contact. Note. The Threshold Days are calendar days.	No	CHAR-TYPE-CD (Characteristic Type Code) CHAR-VAL (Characteristic Type Value) THRES-DAYS (Threshold Days) CC-CLASS (Customer Contact Class) CC-TYPE (Customer Contact Type) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
CASETRAN	CIPQCSTB	This batch process is responsible for calling the algorithm that determines if a case should be transitioned to a new state. Refer to *Automatic Transition Rules* for the details. If **Restrict To Case Type Code** is specified, only cases of this type will be analyzed to determine if they should be transitioned to a new state. If **Restrict To Case Status Code** is specified, only cases in this status will be analyzed to determine if they should be transitioned to a new state. Note, if this parameter is specified, a **Restrict To Case Type Code** must also be defined.	Yes	CASE-TYPE-CD (Restrict To Case Type Code) CASE-STATUS-CD (Restrict To Case Status Code) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
CET	CIPLCETB	The collection event trigger activates all collection events whose trigger date is on or before the supplied business date. Refer to *How Are Collection Events Completed* for more information. Refer to *Calendar vs Work Days* for more information about your date arithmetic options.	Yes	ADD-WORK-DAYS (Y or N) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/60
CLOSEQTE	CIPCUQEB	The close quotes process closes all quotes whose expiration date is on / before the business date. The PROP-SA-ACTION parameter controls whether the proposal contracts linked to the quote's quote details should be marked as declined or cancelled. If you indicate the proposal contracts should be declined, you must also use PROP-DCL-RSN-CD to define the declination reason code to be updated on the contract's.	Yes	PROP-SA-ACTION (DECL or CANC) PROP-DCL-RSN-CD MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
CPM	CIPLCPMB	The collection process monitor removes contracts from collection processes when they have sufficient credits. It will also cancel a collection process when all of its contracts have been removed. Refer to *The C&C Monitors* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/60
DEPINTRF	CIPDINTB	The deposit interest refund process calculates the deposit amount for contracts whose contract type has a special role of "cash deposit".	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
DEPRFND	CIPDRFNB	The deposit refund process refunds deposits to a customer when the customer satisfies the refund criteria.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
DEPRVW	CIPDRVWB	The deposit review process highlights accounts that require an additional deposit. Provide an input Deposit Class to optionally restrict the review to accounts that have contracts belonging to that class.	Yes	DEP-CL-CD (Optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
GLASSIGN	CIPFGLAB	The GL account number assignment process assigns GL account numbers to the GL details associated with financial transactions. Refer to *The GL Interface* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
GLS	CIPFGLEB	The create general ledger download staging process creates a download staging record for every financial transaction that is ready for download. This process populates the FT / Batch Process table with the unique ID of all financial transactions to be interfaced to the general ledger. This process marks each staging record with the GL interface's batch process ID (defined on the installation record). It also stamps the respective batch control's current run number on each record. Note: The GLDL background process uses the information on this staging table to create the consolidated journal entries that are interfaced to your general ledger. Refer to *The GL Interface* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
IB-SPDB	CIPISPDB	This process derives interval data for accounts in the system. Only accounts that have at least one interval contract with derivable profiles linked to it are processed. A 'derivable' profile is an Contract Owned profile where this contract is the owner AND the profile type indicates an "Interval Data Creation" derivation algorithm. Interval data for contracts linked to the Account are derived in Process Priority order as defined on their contract Type. For each contract, the Interval Data Creation algorithms are executed in creation priority order. The standard batch parameter business date will be used by the system to determine until what date to generate data for.	Yes	FORCE-DERIVE-SW (Indicates whether this is a forced derivation - optional.) FRCE-DRV-START-DT (Start date YYYY-MM-DD for forcing derivation - optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	1/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		Use the Force Derive Switch parameter to indicate that you are forcing derivation. Use the Force Derive Start Date to indicate the starting point for the forced derivation.				
IB-STDB	CIPISTDB	This process derives Variance Parameter map data for accounts in the system. Only accounts that have at least one interval contract with derivable Variance Parameter maps linked to it are processed. A 'derivable' map is an Contract Owned map where this contract is the owner AND the map type indicates a "Variance Parameter Data Creation" algorithm. Variance Parameter map data for contracts linked to the Account are derived in Process Priority order as defined on their contract type. For each contract, the Variance Parameter Map Creation algorithms are executed in creation priority order.	Yes	FORCE-DERIVE-SW (Indicates whether this is a forced derivation - optional.) FRCE-DRV-START-DT (Start date YYYY-MM-DD for forcing derivation - optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	1/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		The standard batch parameter business date will be used by the system to determine until what date to generate data for. Use the Force Derive Switch parameter to indicate that you are forcing derivation. Use the Force Derive Start Date to indicate the starting point for the forced derivation.				
IPDSDVB	CIPIPDVB	This process is used to validate interval profile data. It processes interval profiles that were created up to the cutoff date/ time and executes their validation algorithms, if any, defined on their profile type. The algorithms are executed one after the other in their predefined sequence order.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	50/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
IPDSIDB	CIPIPIDB	The Determine Profile For Profile Datasets process attempts to link interval profile data sets to an appropriate profile. It tries to find an interval profile with the same external ID as the one defined on the dataset. Use the Start External ID and End External ID parameters if you only want to process records in that range of Ids. Only profile data sets in pending status that are not already associated with a profile are processed.	Yes	START-EXT-ID (Start External ID - optional) END-EXT-ID (End External ID- optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	50/15
IREGDVB	CIPIRDVB	The Interval Register Data Validation process is used to validate interval register data. It processes interval registers that were created up to the cutoff date/ time and executes their validation algorithms, if any, defined on their interval register type. The algorithms are executed one after the other in their predefined sequence order.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	50/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
IREGIDB	CIPIRIDB	The Determine Register For Register Data Sets process attempts to link interval register data sets to an appropriate interval register. It tries to find an interval register with the same external ID as the one defined on the dataset. Use the Start External ID and End External ID parameters if you only want to process records in that range of Ids. Only interval register data sets in pending status that are not already associated with a register are processed.	Yes	START-EXT-ID (Start External ID - optional) END-EXT-ID (End External ID- optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	50/15
LATEPYMT	CIPBLPCB	The late payment generator creates late payment charges when an account doesn't pay a bill by the end of the grace period. Refer to *How Late Payment Charges Get Calculated* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
NBBAPAY	CIPGACRB	The NBB scheduled payment autopay create process creates autopay records for any scheduled non-billed budget payments where the account is set up for autopay and the non-billed budget is not excluded from autopay.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/ not applicable
NBBPS	CIPGNPSB	The non-billed budget scheduled payment process performs processing for scheduled payment records with a payment date on or before the process business date. If the scheduled payment processing is successful or was not required (i.e. for unmonitored non-billed budgets), the process deletes the current scheduled payment.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
PPAPAY	CIPCPPAB	This process creates automatic payments on the scheduled payment date by calling the automatic payment creation algorithm plugged in on the installation record. Note, automatic payments are only created if: 1) the account has indicated that they pay automatically 2) the payment method on the promise to pay indicates automatic payment should be performed. Note that the automatic payment creation algorithm supplied does not distribute and freeze the automatic payments that are created if the **Autopay Creation Option** on the installation record is set to `Create on Extract Date`. The background process `APAYDSFR` handles this. Refer to *The Big Picture Of Promise To Pay* and *Automatic Payments* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
PPM	CIPCMPPB	This batch process is responsible for monitoring all active payment plans. It looks for payments made by the promise to pay's payor and for contracts in the same debt class as the promise to pay's debt class. It uses these payments to logically offset the promise to pay's scheduled payments. This batch process determines if a promise to pay has been kept, broken or remains active. An ADM trigger is stored for those accounts whose promise to pay have been broken. Refer to *The Promise To Pay Monitor* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
PUPL	CIPPUPLB	The upload payments process creates payment events, payments, and tenders using the records in the various payment staging tables. Refer to *Interfacing Payments From External Sources* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	100/15
PY-RPE	CIPPRPEB	The resolve payments in error process attempts to resolve the following payment errors automatically: A valid account was found but no active contract exists. Refer to *Resolving Exceptions Automatically* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
REACH	CIPBCHRB	This batch process accumulates the total amount paid towards charity contracts in the past year (for each account). If the resultant amount is greater than zero, a temporary bill message will be added to the account. SA-TYPE-CD defines the contract type of your charitable contribution contracts. Note, this is the contract Type code without the Division. All contracts with this contract Type, regardless of Division, will be processed. START-DT is the start date of the financial year in which payments should be accumulated. END-DT is the end date of the financial year in which payments should be accumulated.	No	SA-TYPE-CD. This is the contract type of the charitable contribution contracts. START-DT. It should be entered in the format YYYY-MM-DD. END-DT. This should be entered in the format YYYY-MM-DD.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		BILL-MSG-CD is the code of the bill message that will be added to the customer's bill. Note, your bill message code should have two parameters: the tax year and the total amount of contributions. For example, `Thank you for your charitable contributions in %1 for %2, please keep this bill for tax purposes`. The tax year is derived from the year in the START-DT parameter. ACCT-ID should be zero if this processing should happen for all accounts in the system. If this parameter is non-zero, this process will be limited to the supplied ACCT-ID		BILL-MSG-CD. ACCT-ID MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.		

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		Use ADJ-TYPE-CD1 and CD2 to indicate adjustments whose FTs should be ignored when calculating the contribution amount. This allows you to make adjustments to the charitable contribution contract that are not included in the calculation. (Note: in California, this program is referred to as REACH - Relief for Energy Assistance through Community Help.)		ADJ-TYPE-CD1 ADJ-TYPE-CD2		
REDSAAMT	CIPFFTRB	This process looks for financial transactions linked to each contract that are older than X days (where X is defined on the installation record) that sum to zero. If it finds such FTs, it marks them as "redundant". Redundant FTs do not have to be accessed by the various SQL statements that accumulate an account or contract's balance.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/10

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
SAACT	CIPCSATB	The contract activation process updates pending start and pending stop contracts. Refer to *The System Activates Most Contracts Behind The Scenes* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
SAEXPIRE	CIPCOSVB	The stop expired contract process initiates the stop for all active contracts where the expiration date is on or before the process date. For more information, refer to *Expiring Contracts*.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/ not applicable
SARENEW	CIPCSARB	The contract renewal process renews all active contracts that are due for renewal (i.e. where the renewal date is populated and is less than or equal to the process date). For more information, refer to *Renewing Contracts*.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	300/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
STMPRD	CIPBSTCB	This process creates statements for statement construct records with a pending statement cycle whose processing date has been reached. Refer to *Create Statements* for more information.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
WAITCOM	CIPWTMCB	This workflow waiting process `Completes` a workflow event that has been in the `Waiting` state for longer than X days (X is defined in a parameter supplied to the background process). Refer to *Waiting Events And Their Waiting Processes* for more information.	Yes	WAIT-DAYS MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
WAITMAN	CIPWTMNB	This workflow waiting process `Fails` a workflow event that has been in the `Waiting` state for longer than X days (X is defined in a parameter supplied to the background process). Refer to *Waiting Events And Their Waiting Processes* for more information.	Yes	TIMEOUT-DAYS MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
WAITNT	CIPWTNTB	This workflow waiting process monitors the Notification Upload Staging Response (stored as a context entry) of a notification download staging record that was created as a result of a workflow event. If a Response of Accept appears, the associated event is marked as `Complete`. If a Response of Reject appears, the associated event is marked as `Failed`. This process will `Fail` the workflow event if the event has been in the `Waiting` state for longer the X days (X is defined in a parameter supplied to the background process). Refer to *Waiting Events And Their Waiting Processes* for more information.	Yes	TIMEOUT-DAYS MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
WET	CIPLWETB	The write-off event trigger activates all write-off events whose trigger date is on or before the supplied business date. Refer to *The Big Picture Of Write-Off Events* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
WFET	CIPWWETB	This process does two things: It sets the trigger date of workflow events that are dependent on the completion of earlier workflow events. It activates all workflow events whose trigger date is on or before the supplied business date. If the input Workflow Process Template is populated, only events for workflow processes with that template are processed. Refer to *Workflow Event Dependencies* for more information.	Yes	WF-PROC-TMPL-CD (Workflow Process Template - optional) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
WFPRINIT	CIPWNUSB	The workflow process initiation process creates a workflow process to handle a notification upload staging record. Refer to *How Are Workflow Processes Created* for more information.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15
WPM	CIPLWMOB	The write-off monitor analyzes all accounts with finalized, unpaid contracts. Refer to *The Write Off Monitor* for more information. The input parameter controls how the trigger date is set on write-off events that are created by this process. Refer to *Calendar vs Work Days* for more information about your date arithmetic options.	Yes	ADD-WORK-DAYS MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/15

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Extract Processes

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 can - simply supply the respective run number when you request the batch process.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
APAYACH	CIPPXAPB	The automatic payment ACH (automated clearing house) download extraction process creates the flat file that is interfaced to the ACH. This process downloads all auto pay download staging records associated with its batch control ID that are marked with a supplied run number. If a run number is not supplied, the process extracts all automatic payment download records marked with the current run number. Note: the ACTVTAPY process updates auto pay download records on their extract date so that they will be downloaded by this process. Refer to *Downloading Automatic Payments* for more information.	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
C1-APACH	CIPPXA2B	The automatic payment ACH (automated clearing house) download extraction process creates the flat file that is interfaced to the ACH. This process downloads all auto pay download staging records associated with its batch control ID that are marked with a supplied run number. If a run number is not supplied, the process extracts all automatic payment download records marked with the current run number. The **NBR-DAYS** parameter is added to the scheduled extract date to determine the draft date when the payment amount should be withdrawn from the customer's account. If specified, the value must be a non-negative numeric value. The **BUS-OR-CAL-DAYS** parameter is required when the NBR-DAYS parameter is specified, otherwise it's not allowed. This parameter is used when adding the NBR-DAYS parameter to the scheduled extract date. Valid values are: B - Business days C - Calendar days Note: the ACTVTAPY process updates auto pay download records on their extract date so that they will be downloaded by this process.	No	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed NBR-DAYS= number of days until withdrawal. BUS-OR-CAL-DAYS= business or calendar days. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		Refer to *Downloading Automatic Payments* for more information.			
APDL	CIPADAPB	The A/P download process creates the flat file that is interfaced to your accounts payable software (to cut checks). The process that is delivered has skeletal logic and must be customized by your organization to satisfy the needs of your accounts payable software. In order to adapt the base package program to your specific needs, please following the standard steps: Copy the base package program to your own program. Your own program should be prefixed with the letters `CM` (which stands for "customer modification"). This is important as it prevents the upgrade process from overwriting your new logic. Introduce logic to format the downloaded records into your specific format. If you need assistance, please contact the implementation support group.	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		This process uses all adjustment extract records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process uses all A/P request extract records marked with the current run number. Refer to *The A/P Interface* for more information.			
DWLDCOLL	CIPLXCRB	The collection agency referral download extraction process creates the flat file that contains referrals to be interfaced to your collection agencies. This process extracts all collection agency referral history records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process extracts all referral history records marked with the current run number. The program that is delivered contains skeletal logic that should be used as the basis for your specific processing. The skeletal logic does NOT extract information to a flat file; you must introduce the logic to support your specific flat file format.	No	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		In order to adapt the base package program to your specific needs, please following the standard steps: Copy the base package program to your own program. Your own program should be prefixed with the letters `CM` (which stands for "customer modification"). This is important as it prevents the upgrade process from overwriting your new logic. Introduce logic to format the downloaded records into your specific format. Note: records are written to the referral history table when a collection agency oriented write-off events are activated. Referral history records may also be added manually by an operator. Refer to *How Do Collection Agency Referrals Work?* for more information.			

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
GLDL	CIPFXGLB	The general ledger download process creates the flat file that is interfaced to your general ledger software. This process uses all FT / Batch Process records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process uses all FT / Process records marked with the current run number. In order to adapt the base package program to your specific needs, please following the standard steps: Copy the base package program to your own program. Your own program should be prefixed with the letters `CM` (which stands for "customer modification"). This is important as it prevents the upgrade process from overwriting your new logic. Introduce logic to format the downloaded records into your specific format. Refer to *The GL Interface* for more information.	No	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
LTRPRT	CIPCLTPB	The customer contact letter download process creates the flat file(s) that are interfaced to your letter print software to print letters associated with letter-oriented customer contacts. This process extracts all customer contact records associated with its batch control ID that are marked with a supplied run number. If a run number is not supplied, the process uses all customer contact records associated with its batch control ID that are marked with the current run number. Each downloaded letter's output is written to a filename that is a concatenation of the letter's Letter Template Code and the process's Thread Number. This means that this process can write to multiple files as multiple Letter Template Codes may be downloaded by this process. The information that is extracted and placed on the flat file for each letter is controlled by each customer contact's letter template's extract algorithm. Refer to *Letter Templates Control The Information Merged Onto Letters* for information about how a letter's flat file records are constructed. The **FILE-PATH** parameter controls where the output files are placed.	Yes	*FILE-PATH*= directory path into which output should be placed FIELD-DELIM-SW=Y or N CNTL-REC-SW=Y or N MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		The format of the information on the flat file can be either tilde delimited or in a fixed position (based on the **FIELD-DELIM-SW** parameter). Tilde delimited output is used if you merge the information into a Word template. Fixed position output is used if you merge the information into a Doc 1 template. You can use the **CNTL-REC-SW** parameter to cause the extract to produce a control record that contains batch code, run number, number of letters to print, etc. Refer to *Printing Letters* for more information.			
NDSXTR	CIPWXNDB	The notification download extraction process creates the flat file that is interfaced to your notification routing. This process uses all notification download staging records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process uses all records marked with the current run number. The program that is delivered should be used as a sample as your record formats will differ. In order to adapt the base package program to your specific needs, please following the standard steps:	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		Copy the base package program to your own program. Your own program should be prefixed with the letters `CM` (which stands for "customer modification"). This is important as it prevents the upgrade process from overwriting your new logic. Introduce logic to format the downloaded records into your specific format. Refer to *Downloading Notifications* for more information.			
POSTROUT	CIPBXBLB	The bill print process creates the flat file that is interfaced to your bill print software. This process uses all bill routing extract records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process extracts all bill routing extract records marked with the current run number. The information that is extracted and placed on the flat file for each bill is controlled by each bill route type's extract algorithm. Refer to *Bill Route Controls The Information Merged Onto Bills* for information about how a bill's flat file records are constructed.	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		The **FILE-PATH** parameter controls where the output files are placed. Refer to *Printing Bills* for more information.			
QUOTROUT	CIPCQTXB	The quote-print process creates the flat file that is interfaced to your quote-print software. This process uses all quote-routing extract records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process extracts all quote routing extract records marked with the current run number. The information that is extracted and placed on the flat file for each letter is controlled by each quote route type's extract algorithm. Refer to *Quote Route Type Controls The Information Merged Onto Quotes* for information about how a quote's flat file records are constructed. The **FILE-PATH** parameter controls where the output files are placed. Refer to *Printing Quotes* for more information.	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
RTTYPOST	CIPBRTBB	The bill route type bill print production process invokes the bill extract algorithm defined in the bill route type associated to the batch control. The extract algorithm will determine the bill print software to use and the report or bill template to use to produce all the bill routing records associated with the batch control and run number for a range of bills specified. This process is similar to the Bill Print Production Process (POSTROUT). The differences are: POSTROUT calls the extract algorithm defined in the Bill Route Type for each bill associated to the Bill Route Type; while RTTYPOST only calls the distinct extract algorithm associated with the batch control. POSTROUT produces a flat file that is interfaced to a bill print software, while RTTYPOST calls the bill print software for every distinct extract algorithm found and produce the bills (i.e. the report containing the bills). Refer to *Printing Bills* for more information	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
STMDWLD	CIPBSTXB	The statement download extraction process creates the flat file that contains statement information. This file will be interfaced to your bill print software, or whatever mechanism you will use for sending statements to the appropriate persons. This process extracts all Statement records associated with its batch control that are marked with a supplied run number. If a run number is not supplied, the process extracts all Statement records marked with the current run number. The information that is extracted and placed on the flat file for each statement is controlled by each statement's construct's route type's extract algorithm. Refer to *Statement Route Types Control The Information Merged Onto Statements* for information about how a statement's flat file records are constructed. Refer to *Printing Statements* for more information.	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Adhoc Processes

These are background processes that are run on an ad hoc basis (e.g., if you need to back out bills that were created by the billing process).

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
CPCRALOC	CIPCCCLB	The capital credit allocation process creates allocation service credit events for active capital credit memberships. THIS BACKGROUND PROCESS IS ONLY APPLICABLE TO COMPANIES THAT ISSUE CAPITAL CREDITS TO THEIR CUSTOMERS. Refer to *Allocating Capital Credits* for more information. The program that is delivered one possible method for calculating capital credit allocations. If the logic provided does not meet your business needs, you must adapt the base package program to your specific needs. To do this, please follow the standard steps:	Yes	SC-EVT-TYPE-CD = service credit event type to use for new events SC-EVT-STATUS-FLG = Indicate if the events should be created in pending (10) or active (20) status SCMTY-SUBCAT-NAME = indicate the subcategory to assign to the new event, if applicable ALLOCATION-FACTOR = the allocation factor to use for the calculation of the capital credit allocation BILL-LINE-CHAR-TYPE-SQ = char type identifying bill line containing SQ info	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		Copy the base package program to your own program. Your own program should be prefixed with the letters `CM` (which stands for "customer modification"). This is important as it prevents the upgrade process from overwriting your new logic. Modify the new program as per your business needs. Create a *batch control* record for your new background process, prefixed with `CM`.		BILL-LINE-CHAR-VAL-SQ = char value identifying bill line containing SQ info BILL-LINE-CHAR-TYPE-SALES = char type identifying bill line containing sales (revenue) info BILL-LINE-CHAR-VAL-SALES = char value identifying bill line containing sales (revenue) info FISCAL-YEAR = indicate the fiscal year to assign to the new event		

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
				MIN-CALC-AMT = indicate the minimum calculated amount needed to create an event MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.		
CPCRRETR	CIPCCCRB	The capital credit retirement process creates retirement service credit events for active capital credit memberships. THIS BACKGROUND PROCESS IS ONLY APPLICABLE TO COMPANIES THAT ISSUE CAPITAL CREDITS TO THEIR CUSTOMERS.	Yes	SC-EVT-TYPE-CD = service credit event type to use for new events SC-EVT-STATUS-FLG = Indicate if the events should be created in pending (10) or active (20) status	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		Refer to *Capital Credit Retirement* for more information. The program that is delivered one possible method for calculating capital credit retirement amounts. If the logic provided does not meet your business needs, you must adapt the base package program to your specific needs. To do this, please follow the standard steps: Copy the base package program to your own program. Your own program should be prefixed with the letters `CM` (which stands for "customer modification"). This is important as it prevents the upgrade process from overwriting your new logic. Modify the new program as per your business needs. Create a *batch control* record for your new background process, prefixed with `CM`.		PCT-ALLOC-TO-RETIRE = Percentage of balance to retire SCMTY-SUBCAT-NAME = indicate the subcategory to limit retirement to a single subcategory (if blank, all subcategories are processed) CTRL-ID = stamped onto new events - used for restart control MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.		

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
F1-AVALG	Java	This process regenerates algorithm type and their related algorithm information to be displayed by the Application Viewer. It produces a series of XML files in a designated folder under the application viewer /data folder. Refer to *Application Viewer Generation* for more information on this background process is used.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	NA
F1-AVMO	Java	This process regenerates maintenance object information to be displayed by the Application Viewer. It reads the meta-data maintenance object information and produces a series of XML files in a designated folder under the application viewer /data folder. Refer to *Application Viewer Generation* for more information on this background process is used.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
F1-AVTBL	Java	This process regenerates table and column information to be displayed by the Application Viewer. It reads the meta-data table and related entities and produces a series of XML files in a designated folder under the application viewer /data folder. Refer to *Application Viewer Generation* for more information on this background process is used.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
MASSCNCL	CIPBMCNB	The mass bill cancellation process removes an entire batch of bills that were created by the BILLING process. Refer to *Canceling A Batch Of Bills After They're Complete* for more information.	Yes	BILL-CYC-CD= the bill cycle associated with the bills WIN-START-DT=the bill cycle window start date associated with the bills. This should be entered in the format YYYY-MM-DD. BILL-DT= the date on which the bills were created. This should be entered in the format YYYY-MM-DD. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	100/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
MASSROBL	CIPBMROB	The mass bill reopen process reopens an entire batch of bills that were completed by the BILLING process. Refer to *Reopening A Batch Of Bills After They're Complete* for more information.	Yes	BILL-CYC-CD= the bill cycle associated with the bills WIN-START-DT= the bill cycle window start date associated with the bills BILL-DT= the date on which the bills were created. This should be entered in the format YYYY-MM-DD. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	Yes	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
NEWLANG	CIPZLNGB	This Create New Language batch program duplicates all language specific rows from the source to the target language. Once this program has run you will need to translate the language specific columns. This program can be rerun at anytime. It will only duplicate entries where they do not already exist. Note if you run this program with the "DELETE" action then the source and target language must be the same. If you have any questions, please see your implementation team for more information.	Yes	SOURCE-LANGUAGE= source language code TARGET-LANGUAGE= target language code PROGRAM-ACTION= (DELETE or INSERT) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	No	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
UPDERR	CIPZUESB	The process updates `In Progress` threads in an abnormally terminated batch run to be `Error`. When at least one thread is in `Error`, this process also updates the status of the batch run to be `Error`. Refer to *Dealing With Abnormally Terminated Background Processes* for information about when and why this process would be executed.	No	BATCH-CD-IN-PROGRESS = the batch control ID of the abnormally terminated batch process BATCH-NBR-IN-PROGRESS = the run number of the abnormally terminated batch process UPD-ALL-THRDS-SW must be Y or N. A value of Y means that the status of all `In Progress` threads will be changed to `Error`. A value of N means that the next parameter must be supplied.	No	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Error Generates To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
				BATCH-THRD-NBR-IN-PROGRESS = the thread number whose status should be changed from `In Progress` to `Error`. This parameter should only be specified if UPD-ALL-THRDS-SW = N. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.		

Please refer to *Column Descriptions* for more information on the columns used in the table above.

To Do Entry Processes

These are background processes whose main purpose is to generate To Do Entry records based on a certain condition. Refer to *To Do Entries Created By Background Processes* for the details. The section that appears below simply lists these processes.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-BCUPL	CIPQBCEB	This background process creates a To Do entry for every billable charge upload record that's in error.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-BIERR	CIPQBIEB	This background process creates a To Do entry for every bill that's in error.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-BSERR	CIPQBSEB	This background process creates a To Do entry for every bill segment that's in error.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-BTERR	CIPQBERB	This background process creates To Do Entry for any other batch processes that ended in error. A To Do Entry is only created if one does not already exist.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-CCCB	CIPQCCCB	This background process creates a To Do entry for customer contacts that have been flagged to generate a To Do entry on a future date. Note well, most To Do background processes create To Do entries in the `pending` state. If the customer contact indicates a specific user should be notified (as opposed to notifying a group of users - a role), the To Do entry will be created in the `being worked` state and it will be assigned to the designated user.	Yes	LEAD-DAYS = Number of days before the customer contact's reminder date that the To Do entry should be created. Valid values of 0 to 99 are acceptable. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-CEVT	CIPQCEVB	This background process creates a To Do entry for collection events that should generate a To Do entry.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-CLERR	CIPQCLEB	This background process creates a To Do Entry for any batch process that has root objects that created an error. A To Do Entry is only created if one does not already exist.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-DTCST	CIPQDTCB	This background process creates a To Do entry for deposit control staging / tender control staging records that are in error.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-ECBK	CIPQENHB	This background process creates To Do entries for held orders.	Yes	DAYS-BEF-CALLBK = Number of days before the order's reminder date that the To Do entry should be created MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-EPND	CIPQENPB	This background process creates To Do entries for pending orders.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-MODTL	CIPQODMB	This background process creates a To Do entry for every disputed match event	Yes	NO-OF-DAYS = Number of days old the match event must be before a To Do entry is created (this prevents young entries from appearing on To Do lists) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-MONTL	CIPQONMB	This background process creates a To Do entry for every open, non-disputed match event	Yes	NO-OF-DAYS = Number of days old the match event must be before a To Do entry is created (this prevents young entries from appearing on To Do lists) MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-NCDEX	CIPQNCDB	This background process creates a To Do entry for every non-cash deposit that is due to expire within the next X days (X is a parameter). Note. The process checks for completed To Do entries for this To Do Type and Account Id and will not create a new To Do if a completed one exists.	Yes	DAYS-TO-EXPIRY = Number of days prior to NCD expiration that the entry should be created MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-NOBC	CIPQNBCB	This background process creates a To Do entry for every account that doesn't have a bill cycle but has active contracts.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-NTDWN	CIPQNTDB	This background process creates a To Do entry for every notification download staging record that's in error.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-NTUPL	CIPQNTUB	This background process creates a To Do entry for every notification upload staging record that's in error.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-PYERR	CIPQPAYB	This background process creates a To Do entry for every payment that's in error or that is unfrozen.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-PYUPL	CIPQPYUB	This background process creates a To Do entry for every payment staging record that's in error.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-SSFTL	CIPQSSLB	This background process creates a To Do entry for pending start/ stops that are older than the number of days specified. This catches start/stop requests that have gone unfulfilled.	Yes	NO-OF-DAYS = the number of days old a pending start / stop contract must be to be considered unfulfilled. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-UNBAL	CIPQPYEB	This background process creates a To Do entry for every payment event that's unbalanced.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-WEXTL	CIPQWEXB	This background process creates a To Do entry for every workflow event that's in error.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-WOEVT	CIPQWEVB	This background process creates a To Do entry for write-off events that should generate a To Do entry.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-XAIDN	CIPQXADB	This background process creates a To Do entry for every XAI Download Staging exception.	No	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
TD-XAIUP	CIPQXAUB	This background process creates a To Do entry for every XAI Upload Staging in error.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Object Validation Processes

These background processes are run to validate the master data objects. These programs are typically only run as part of the conversion and upgrade processes.

Note: Another use for these programs. In addition to validating your objects after conversion or an upgrade, the validation programs listed below have another use. Say for example, you want to experiment with changing the validation of a person and you want to determine the impact of this new validation on your existing persons. You could change the validation and then run the person validation object - it will produce errors for each person that fails the new validation.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-ACCT	CIPVACCB	Validate the account object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	**Program Name**	**Description**	**Multiple Threads**	**Extra Parameters**	**Records Between Commits / Minutes Between Cursor Re-Initiation**
VAL-BCHG	CIPVBCGB	Validate the billable charge object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. STATUS1=validate rows with this status STATUS2=validate rows with this status MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-CEVT	CIPVCEVB	Validate the contract option event object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	**Program Name**	**Description**	**Multiple Threads**	**Extra Parameters**	**Records Between Commits / Minutes Between Cursor Re-Initiation**
VAL-COLL	CIPVCLPB	Validate the collection process object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. STATUS1=validate rows with this status STATUS2=validate rows with this status MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-COP	CIPVCOPB	Validate the contract option object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	**Program Name**	**Description**	**Multiple Threads**	**Extra Parameters**	**Records Between Commits / Minutes Between Cursor Re-Initiation**
VAL-PER	CIPVPERB	Validate the person object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-SA	CIPVSVAB	Validate the contract object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	**Program Name**	**Description**	**Multiple Threads**	**Extra Parameters**	**Records Between Commits / Minutes Between Cursor Re-Initiation**
VAL-SCM	CIPVSCMB	Validate the service credit membership object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-TDS	CIPVTDSB	Validate the Variance Parameter data set object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	**Program Name**	**Description**	**Multiple Threads**	**Extra Parameters**	**Records Between Commits / Minutes Between Cursor Re-Initiation**
VAL-TMAP	CIPVTMBB	Validate the Variance Parameter map object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-WFP	CIPVWPRB	Validate the workflow process object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. STATUS1=validate rows with this status STATUS2=validate rows with this status MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
VAL-WOP	CIPVWOPB	Validate the write off process object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value OVRD-HIGH-ID=key value to override the calculated end-key value SKIP-ROWS= nth row to be processed, for example 10 to process every 10 th row. STATUS1=validate rows with this status STATUS2=validate rows with this status MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Referential Integrity Validation Processes

The following table lists every background process that validates transaction data. These programs are typically run as part of the conversion and upgrade processes.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVAAPV	CIPVRNVB	Foreign Key validation for Account Automatic Payment	No	None	N/A
CIPVACHV	CIPVRNVB	Foreign Key validation for Account Characteristics	No	None	N/A
CIPVACPV	CIPVRNVB	Foreign Key validation for Account Person Relationship	No	None	N/A
CIPVADJV	CIPVRNVB	Foreign Key validation for Adjustment	No	None	N/A
CIPVAPRV	CIPVRNVB	Foreign Key validation for A/P Check Request	No	None	N/A
CIPVARHV	CIPVRNVB	Foreign Key validation for Collection Agency Referral History	No	None	N/A
CIPVARSV	CIPVRNVB	Foreign Key validation for Credit Review Schedule	No	None	N/A
CIPVBCHV	CIPVRNVB	Foreign Key validation for Bill Characteristic	No	None	N/A
CIPVBCGV	CIPVRNVB	Foreign Key validation for Billable Charge	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVBCLV	CIPVRNVB	Foreign Key validation for Billable Charge Line	No	None	N/A
CIPVBFVV	CIPVRNVB	Foreign Key validation for Bill Factor Value	No	None	N/A
CIPVBLLV	CIPVRNVB	Foreign Key validation for Bill Header	No	None	N/A
CIPVBLMV	CIPVRNVB	Foreign Key validation for Bill Messages	No	None	N/A
CIPVBLRV	CIPVRNVB	Foreign Key validation for Bill Routing	No	None	N/A
CIPVBSAV	CIPVRNVB	Foreign Key validation for Bill - Contract Balance Snapshot	No	None	N/A
CIPVBSCV	CIPVRNVB	Foreign Key validation for Bill Segment Calc Header	No	None	N/A
CIPVBSIV	CIPVRNVB	Foreign Key validation for Bill Segment Item	No	None	N/A
CIPVBSLV	CIPVRNVB	Foreign Key validation for Bill Segment Calc Line	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVCARV	CIPVRNVB	Foreign Key validation for Collection Agency Referral	No	None	N/A
CIPVCCFV	CIPVRNVB	Foreign Key validation for Contract Option Characteristic	No	None	N/A
CIPVCECV	CIPVRNVB	Foreign Key validation for Collection Event/ Customer Contact	No	None	N/A
CIPVCLPV	CIPVRNVB	Foreign Key validation for Collection Process	No	None	N/A
CIPVCLSV	CIPVRNVB	Foreign Key validation for Collection Process Contract	No	None	N/A
CIPVCOLV	CIPVRNVB	Foreign Key validation for Contract Option Language	No	None	N/A
CIPVCRTV	CIPVRNVB	Foreign Key validation for Credit Rating History	No	None	N/A
CIPVCSCV	CIPVRNVB	Foreign Key validation for Customer Contact	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVCVCV	CIPVRNVB	Foreign Key validation for Cont Opt Event Characteristic	No	None	N/A
CIPVCVNV	CIPVRNVB	Foreign Key validation for Collection Event	No	None	N/A
CIPVFTFV	CIPVRNVB	Foreign Key validation for Financial Transaction	No	None	N/A
CIPVFTGV	CIPVRNVB	Foreign Key validation for Fin'l Transaction Gen Ledger	No	None	N/A
CIPVFTPV	CIPVRNVB	Foreign Key validation for Fin'l Transaction Process	No	None	N/A
CIPVINLV	CIPVRNVB	Foreign Key validation for Interval Profile Lang	No	None	N/A
CIPVMSGV	CIPVRNVB	Foreign Key validation for Account Bill Messages	No	None	N/A
CIPVNBSV	CIPVRNVB	Foreign Key validation for Non-billed Budget Contract	No	None	N/A
CIPVNCDV	CIPVRNVB	Foreign Key validation for Non Cash Deposit	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVNPMV	CIPVRNVB	Foreign Key validation for NBB Contract Payment Schedule Parameter Values	No	None	N/A
CIPVNSPV	CIPVRNVB	Foreign Key validation for NBB Contract Scheduled Payments	No	None	N/A
CIPVPAOV	CIPVRNVB	Foreign Key validation for Person Address Override	No	None	N/A
CIPVPAYV	CIPVRNVB	Foreign Key validation for Payment Header	No	None	N/A
CIPVPIDV	CIPVRNVB	Foreign Key validation for Person Identifier	No	None	N/A
CIPVPNMV	CIPVRNVB	Foreign Key validation for Person Name	No	None	N/A
CIPVPPEV	CIPVRNVB	Foreign Key validation for Person to Person	No	None	N/A
CIPVPPHV	CIPVRNVB	Foreign Key validation for Person Phone	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVPRCV	CIPVRNVB	Foreign Key validation for Person Characteristics	No	None	N/A
CIPVPSAV	CIPVRNVB	Foreign Key validation for Person Seasonal Address	No	None	N/A
CIPVPSGV	CIPVRNVB	Foreign Key validation for Payment Segment	No	None	N/A
CIPVPYCV	CIPVRNVB	Foreign Key validation for Payment Characteristics	No	None	N/A
CIPVREFV	CIPVRNVB	Foreign Key validation for Interval Register Data	No	None	N/A
CIPVREGV	CIPVRNVB	Foreign Key validation for Register	No	None	N/A
CIPVRGCV	CIPVRNVB	Foreign Key validation for Register Characteristics	No	None	N/A
CIPVRGRV	CIPVRNVB	Foreign Key validation for Register Read	No	None	N/A
CIPVSACV	CIPVRNVB	Foreign Key validation for Contract Characteristics	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVSAHV	CIPVRNVB	Foreign Key validation for Contract Rate Schedule History	No	None	N/A
CIPVSAOV	CIPVRNVB	Foreign Key validation for Contract Contract Terms	No	None	N/A
CIPVSAPV	CIPVRNVB	Foreign Key validation for Contract SP	No	None	N/A
CIPVSAQV	CIPVRNVB	Foreign Key validation for Contract Contract Quantity	No	None	N/A
CIPVSARV	CIPVRNVB	Foreign Key validation for Contract Recurring Charge	No	None	N/A
CIPVSCAV	CIPVRNVB	Foreign Key validation for SCM Account	No	None	N/A
CIPVSCCV	CIPVRNVB	Foreign Key validation for SCM Characteristic	No	None	N/A
CIPVSCFV	CIPVRNVB	Foreign Key validation for Service Credit Event FT	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVSCOV	CIPVRNVB	Foreign Key validation for Contract Override Contract Option	No	None	N/A
CIPVSCPV	CIPVRNVB	Foreign Key validation for Contract Option	No	None	N/A
CIPVSCVV	CIPVRNVB	Foreign Key validation for Service Credit Event	No	None	N/A
CIPVSEGV	CIPVRNVB	Foreign Key validation for Bill Segment	No	None	N/A
CIPVSMGV	CIPVRNVB	Foreign Key validation for Contract Message	No	None	N/A
CIPVSMIV	CIPVRNVB	Foreign Key validation for Multi Item	No	None	N/A
CIPVSQTV	CIPVRNVB	Foreign Key validation for Bill Segment Service Quantity	No	None	N/A
CIPVSSCV	CIPVRNVB	Foreign Key validation for Contract SP Characteristics	No	None	N/A
CIPVSTMV	CIPVRNVB	Foreign Key validation for Contract Variance Parameter Map	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVTBVV	CIPVRNVB	Foreign Key validation for Variance Parameter Bill Factor Value	No	None	N/A
CIPVTCVV	CIPVRNVB	Foreign Key validation for Variance Parameter Contract Value	No	None	N/A
CIPVTMLV	CIPVRNVB	Foreign Key validation for Variance Parameter Map Lang	No	None	N/A
CIPVTNCV	CIPVRNVB	Foreign Key validation for Payment Tender Characteristics	No	None	N/A
CIPVTNDV	CIPVRNVB	Foreign Key validation for Payment Tender	No	None	N/A
CIPVTOFV	CIPVRNVB	Foreign Key validation for Variance Parameter Data	No	None	N/A
CIPVTRNV	CIPVRNVB	Foreign Key validation for Trend	No	None	N/A
CIPVWECV	CIPVRNVB	Foreign Key validation for Workflow Event Context	No	None	N/A

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CIPVWEDV	CIPVRNVB	Foreign Key validation for Workflow Event Dependency	No	None	N/A
CIPVWEVV	CIPVRNVB	Foreign Key validation for Workflow Event	No	None	N/A
CIPVWOCV	CIPVRNVB	Foreign Key validation for Write Off Event/ Customer Contact	No	None	N/A
CIPVWOPV	CIPVRNVB	Foreign Key validation for Write Off Process	No	None	N/A
CIPVWOSV	CIPVRNVB	Foreign Key validation for Write Off Process/ Contract	No	None	N/A
CIPVWOVV	CIPVRNVB	Foreign Key validation for Write Off Event	No	None	N/A
CIPVWPCV	CIPVRNVB	Foreign Key validation for Workflow Process Context	No	None	N/A
CIPVWPRV	CIPVRNVB	Foreign Key validation for Workflow Process	No	None	N/A

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Conversion Processes

These background processes are run only when converting or migrating data from external applications into the system. Your company may never use them depending upon your data migration strategy.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CNV-ADM	CIPVADMB	Creates ADM triggers for converted accounts.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
CNV-BAL	CIPVVSAB	Sets the correct balance for all contracts created during conversion. It creates adjustments that cause each contract's current and payoff balances to equal their balance in the prior system.	No	SET-PYOF-ZERO= Adjustment Type to set payoff balance to zero SET-CURR-ZERO = Adjustment Type to make current balance zero SET-PYOF-BAL= Adjustment Type to set payoff balance to the desired balance SET-CURR-BAL = Adjustment Type to set current balance to the desired balance and age	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
				BASE-DATE= Base Date of Aged Debt. This should be entered in the format YYYY-MM-DD. *FILE-PATH*= Path and filename of input data file MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	
CNV-BCG	CIPVCBCB	This process resets the Balance Control column on all FT's so that the FT's can be included in a balance control (see the last step below) after they have been transferred to production.	Yes	MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Conversion Processes Executed In The Staging Database

There are many other background processes that are only executed if you use the conversion tool to load historical data into your production database. These programs perform the following tasks:

- **Key Assignment Programs.** Background processes of this type assign random, clustered keys to the rows in the staging database.
- **Insertion Programs.** Background processes of this type insert converted rows into production from the staging database.

Purge Processes

These background processes are used to purge historical records from certain objects that generate a large number of entries and may become unwieldy over time.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
BCUP-PRG	CIPCDBSB	Purges `completed` billable charge upload objects.	Yes	NO-OF-DAYS = number of days after the creation date that a `completed` billable charge upload object should be purged MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15
PYUP-PRG	CIPPDUSB	Purges `completed` tender upload objects.	Yes	NO-OF-DAYS = number of days after the related tender's payment event's creation date that a `completed` tender upload object should be purged MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
TD-PURGE	CIPQDTDB	Purges `completed` To Do entries.	Yes	NO-OF-DAYS = number of days after the completion date that a `completed` To Do entry should be purged DEL-ALL-TD-SW = Y or N. If this switch is Y, all `completed` To Do entries that are old enough will be deleted. If N, the next parameter defines the specific type of To Do entry that will be deleted. DEL-TD-TYPE-CD. This parameter is only used if DEL-ALL-TD-SW is N. It contains the To Do type code whose `completed` entries will be deleted. MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
XMLUP-PR	CIPXDXUB	Purges `completed` XML upload objects.	Yes	NO-OF-DAYS = number of days after the completion date that a `completed` XML upload object should be purged MAX-ERRORS = Used to override the maximum number of errors after which the batch must be terminated.	200/15

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Column Descriptions

The following descriptions explain the parameters used in the above tables:

- **Batch Control ID.** As described earlier, every background process has an associated batch control record. This column contains the unique identifier of each process' batch control record.
- **Program Name.** This is the name of the program.
- **Description.** This column describes each background process.
- **Multiple Threads.** This column indicates if the background process uses the thread number and thread count to control parallel processing. For more information, refer to the **Parameters Supplied to Background Processes** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Background Processes** > **Understanding Background Processes**).
- **Extra Parameters.** This column indicates if the background process uses additional parameters. For more information, refer to the **Parameters Supplied to Background Processes** section (under **Oracle Revenue Management and Billing** > **Administrative Processes** > **Framework Administrative User Guide** > **Background Processes** > **Understanding Background Processes**).
- **Error Generates To Do.** This column indicates if the background process generates a To Do entry for object-specific errors as described in *Processing Errors*.
- **Records Between Commits / Minutes Between Cursor Re-Initiation.** These values represent the maximum number of records between commits to the database and the number of minutes between cursor re-initiations. The process will issue a commit whenever the maximum records threshold has been exceeded. And, whenever a commit is issued, the process checks if the number of minutes between cursor initiation has been exceeded and if so, it will re-initiate the cursor. These values may be overridden when a specific background process is submitted. For more information, refer to the **Parameters Supplied to Background Processes** section (under

Oracle Revenue Management and Billing > **Administrative Processes** > **Framework Administrative User Guide** > **Background Processes** > **Understanding Background Processes**).

Batch Process Dependencies

The contents of this section illustrate the periodicity and dependencies between the various background processes described above.

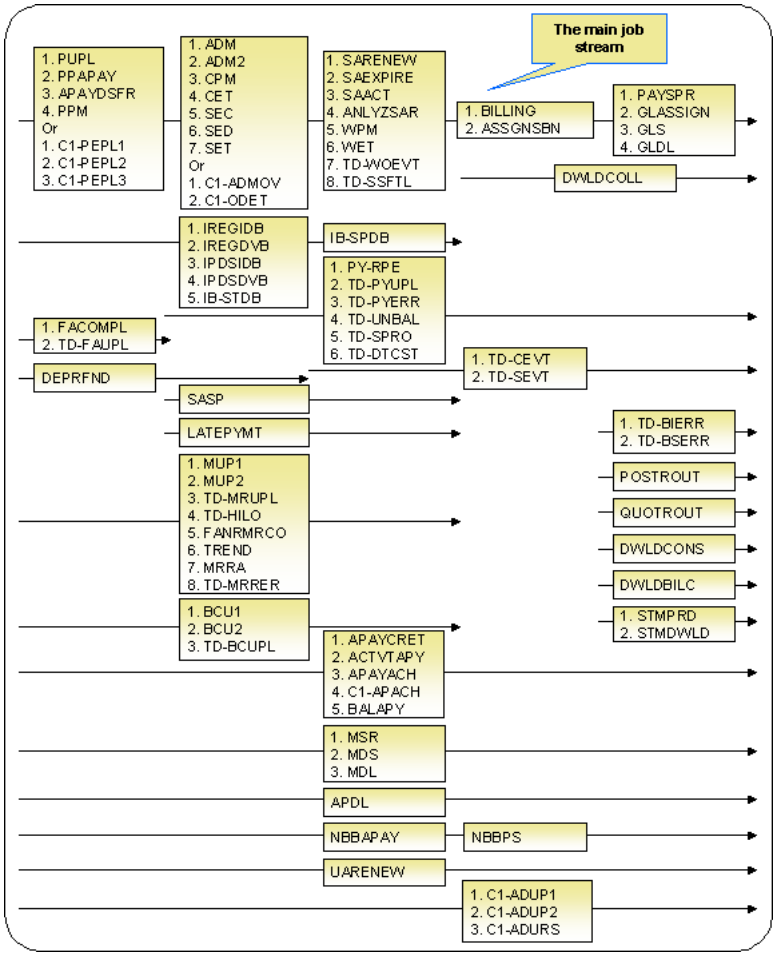
Batch Schedulers and Return Codes

If you use a batch scheduler (e.g., Control-M, Tivoli) to control the execution of your batch processes, it will be interested in the possible values of each process's return code. The return code is a number that indicates if the process ended successfully. All product processes will return one of the following return code values:

- `0` (zero). A value of zero means the batch process ended normally.
- `2`. A value of 2 means the batch process detected a fatal error and aborted.

The Nightly Processes

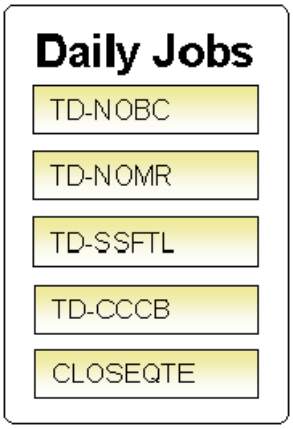
The following diagram illustrates the dependencies between the batch processes.



The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially. When multiple boxes exist on a timeline, all processes in an earlier

box must execute before the subsequent box is executed. Those timelines that appear beneath the Main Job Stream's timeline indicate when the timeline's respective processes can be executed in respect of the Main Job Stream.

The following diagram illustrates the daily batch processes for which there are no dependencies.

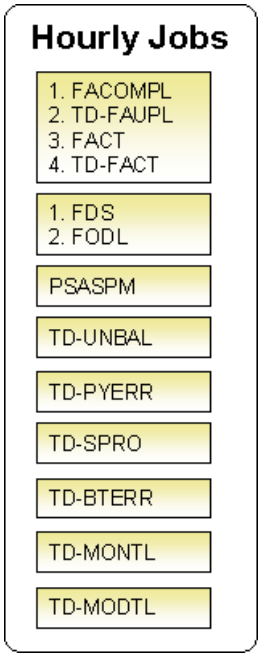


The mnemonics in the boxes refer to the individual batch processes described above.

Note: No dependencies exist. As you can see, there are no dependencies between the boxes (meaning they may be run in parallel).

The Hourly Processes

The following diagram illustrates the dependencies between the hourly batch processes.

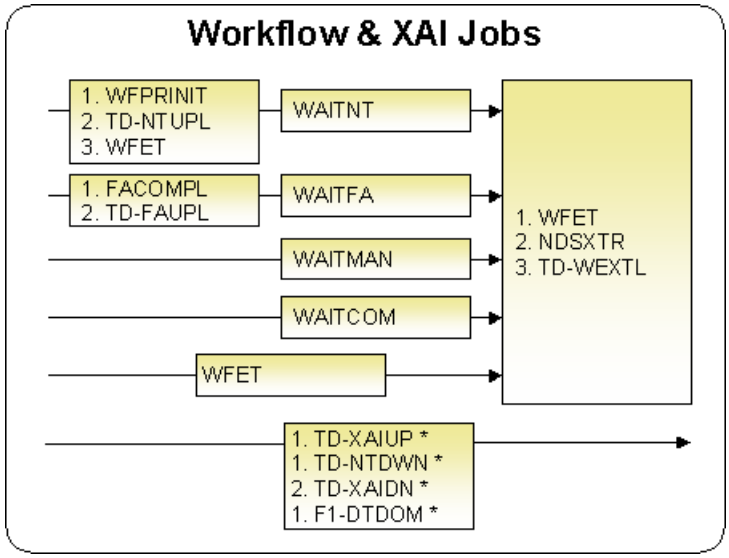


The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially.

Note: No dependencies exist. As you can see, there are no dependencies between the boxes (meaning they may be run in parallel).

The Workflow and XAI Processes

The following diagram illustrates the dependencies between the workflow and XAI background processes. While these processes should be run at least once a day, you may want to consider running them more frequently (depending on how frequently you interface notifications into the system).



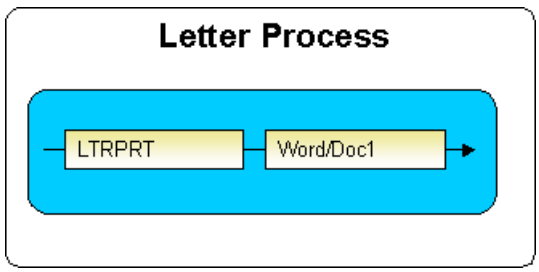
The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially. When multiple boxes exist on a timeline, all processes in an earlier box must execute before the subsequent box is executed.

* These processes create and/or clean up To Do entries for XAI upload staging, notification download staging, XAI download staging records or outbound messages in error. They are only applicable if your organization is using the XAI tool because only the XAI tool will mark one of these records in error.

The Letter Processes

To extract information for your various letters, only one background process, LTRPRT, is required regardless of the different types of letters you have. This process simply calls an algorithm plugged-in on the respective letter template to construct its flat-file content.

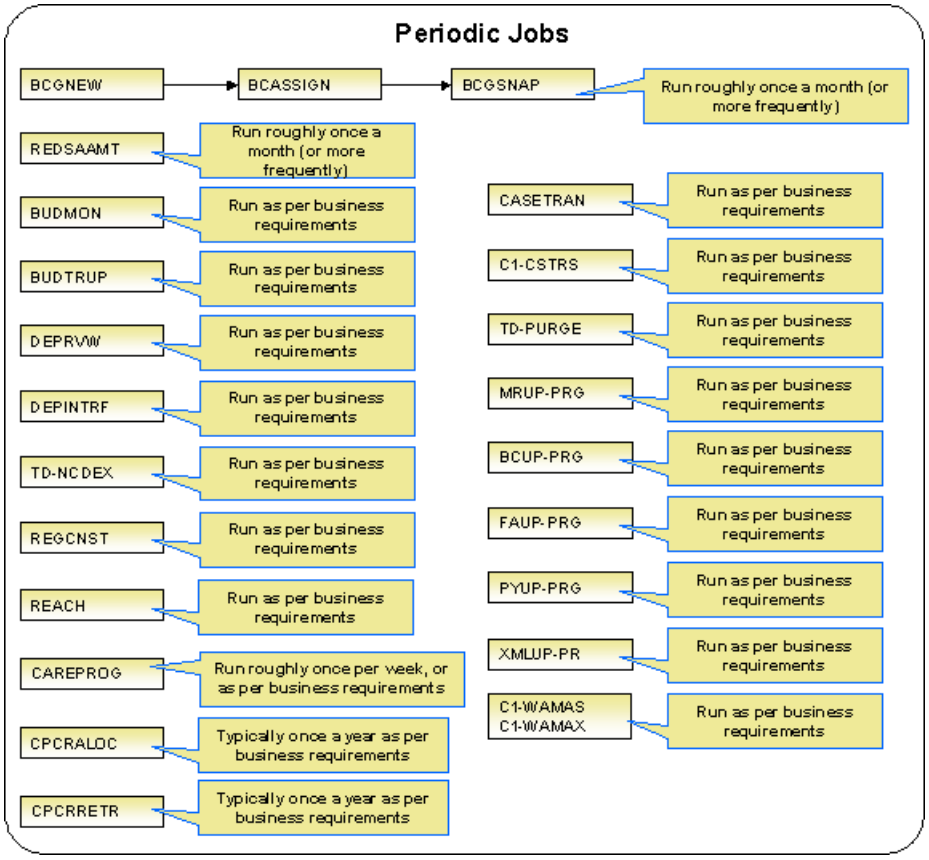
The following diagram illustrates the dependencies for the letter background process. While this process should be run at least on a daily basis, you may want to consider running it more frequently (depending on how frequently you produce letters).



The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially. When multiple boxes exist on a timeline, all processes in an earlier box must execute before the subsequent box is executed.

The Periodic Processes

The following diagram illustrates the dependencies between the periodic background processes. While many of these processes should be run at least on a monthly basis, you may want to consider running them more frequently (depending on business requirements).



The mnemonics in the boxes refer to the individual batch processes described above.

Note: Few dependencies exist. As you can see, there are few dependencies between the boxes (meaning they may be run in parallel).

How To Set Up A New Extract Process

Several background processes delivered with the system are used to interface information out of the system. The topics in this section describe when and how to introduce an additional extract process.

Setting Up Automatic Payment Extracts

You will need an automatic payment extract for every mechanism your company uses to route automatic payment requests to a financial institution / clearing house. For example:

- You will need an automatic payment extract to interface records to the Automated Clearing House (ACH) if you allow customer to pay via credit card or direct debit from a checking account. The ***APAYACH*** and `C1-APACH` processes delivered with the system are intended to be used to handle this function.

If you need additional automatic payment extract processes, set up the following information:

- Add a new *batch control* record. Populate the fields as follows:
  - **Batch Process.** Assign an easily recognizable unique ID for the automatic payment extract process.
  - **Description.** Enter a description of the automatic payment extract process.
  - **Accumulate All Instances.** Turn this switch on.
- Use *Auto Pay Route Type* to define the auto pay extract process to be used for each route type.

The Big Picture of Sample & Submit Request

Sample and Submit refers to the ability to create Activity Requests. This is functionality that enables an implementation team to design an ad-hoc batch process using the configuration tools.

Some examples of such processes are:

- Send a letter to customers that use credit cards for auto pay and the credit card expiration date is within 30 days of the current date.
- Stop auto pay for customers that use credit cards as the form of payment if the credit card has already expired. Notify the customer that their auto pay agreement has been terminated and that they need to call to reinstate.
- Select auto pay accounts that have more than X non-sufficient fund penalties, stop the auto pay agreement and notify the customer.

Note: The terms activity request and sample & submit request may be used interchangeably.

Activity Type

For each type of process that your implementation wants to implement, you must configure an activity type to capture the appropriate parameters needed by the activity request.

Preview a Sample Prior to Submitting

To submit a new activity request, a user must select the appropriate activity type and enter the desired parameter values, if applicable. After entering the parameters, the following actions are possible:

- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can **Save** to submit the request, go **Back** to adjust the parameters or **Cancel** the request.
- Click **Cancel** to cancel the request.
- Click **Save** to skip the preview step and submit the request.

When an activity request is saved, the job is not immediately submitted for real time processing. The record is save in the status **Pending** and a monitor process for this record's business object is responsible for transitioning the record to **Complete**.

As long as the record is still **Pending**, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

The actual work of the activity request, such as generating customer contact records to send letters to a set of customers, is performed when transitioning to **Complete** (using an enter processing algorithm for the business object).

Credit Card Expiration Notice

The base product supplies a sample process to find customers that use credit cards for auto pay and the credit card expiration date is within x days of the current date.

To this functionality the following configuration tasks are needed:

- Define an appropriate customer contact class and type to use.
- Define appropriate activity request Cancellation Reasons. Cancellation reasons are defined using a customizable lookup. The lookup field name is **C1_AM_CANCEL_RSN_FLG**.
- Define an activity type for the business object **C1-NotifyExpiringCreditCardTyp**. You may define default parameter values for the number of days for expiration and customer contact class and type.

Exploring Activity Request Data Relationships

Use the following links to open the application viewer where you can explore the physical tables and data relationships behind the activity request functionality:

- Click C1-ACM-ACTTY to view the activity type maintenance object's tables.
- Click C1-ACM-ACTRQ to view the activity request maintenance object's tables.

Defining a New Activity Request

To design a new ad-hoc batch job that users can submit via Sample and Submit, first create a new Activity Type business object. The base product BO for activity type **C1-NotifyExpiringCreditCardTyp** may be used as a sample.

The business object for the activity request includes the functionality for selecting the records to process, display a preview map for the user to review and to perform the actual processing. The base product BO for activity request **C1-NotifyExpiringCreditCardReq** may be used as a sample. The following points highlight the important configuration for this business object:

- Special BO options are available for activity request BOs to support the Preview Sample functionality.
    - Activity Request Preview Service Script. This script is responsible for retrieving the information displayed when a user asks for a preview of a sample of records.
    - Activity Request Preview Map. This is the map that is invoked to display the preview sample results.
- The enter algorithm plugged into the **Complete** state is responsible for selecting all the records that satisfy the criteria and processing the records accordingly.

Setting Up Activity Types

Activity types define the parameters to capture when submitting an activity request via Sample and Submit. To set up an activity type, open **Admin Menu**, **Activity Type**.

The topics in this section describe the base-package zones that appear on the **Activity Type** screen.

Activity Type List

The **Activity Type List** zone lists every activity type. The following functions are available:

- Click the broadcast button to open other zones that contain more information about the adjacent activity type.
- Click the **Add** link in the zone's title bar to add a new activity type.

Activity Type

The Activity Type zone contains display-only information about an activity type. This zone appears when an activity type has been broadcast from the Activity Type List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click the **Edit** button to start a business process that updates the activity type.
- Click the **Delete** button to start a business process that deletes the activity type.
- Click the **Duplicate** button to start a business process that duplicates the activity type.
- State transition buttons are available to transition the activity type to an appropriate next state.

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

Maintaining Sample & Submit Requests

Use the Sample and Submit transaction to view and maintain pending or historic activity requests. Navigate using **Main Menu**, **Batch**, **Sample & Submit Request**.

Sample & Submit Request Query

Use the query portal to search for an existing sample & submit request. Once a request is selected, you are brought to the maintenance portal to view and maintain the selected record.

Sample & Submit Request Portal

This portal appears when a sample & submit request has been selected from the **Sample & Submit Request Query** portal.

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

Sample & Submit

The **Sample & Submit** zone contains display-only information about an activity (sample & submit) request. The following functions are available:

- Click the **Edit** button to modify the parameters. Refer to *Preview a Sample Prior to Submitting*on page 477 for more information.
- If the activity request is in a state that has valid next states, buttons to transition to each appropriate next state are displayed.

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

Sample & Submit Log

This is a standard log zone.

Chapter 9

Defining Service Credit Options

Topics:

Some companies allow their customers to participate in a special rewards program. The term "service credits" is used to describe a program that rewards customers for their business. The topics in this section provide details to help you set up the control tables required to support any service credit program that your company supports.

Note: Penalty Points. The service credits functionality is described in the documentation with the assumption that it is used for accumulating points to reward your customers. However, if your company has a business need to record penalty points for a customer, the service credits functionality may be used for that purpose as well.

The Big Picture Of Service Credits

The topics in this section provide background information about service credit functionality.

Service Credit Membership

Let's look at some examples of special programs that may use service credits functionality:

- Capital credits - When customers receive their services from a financial service company, they are considered "members" of the financial service company and may over time receive capital credit allocations from the financial service company based on their service history and the financial service company's profits allocated during that time period.
- Frequent flier miles - Perhaps your company has made an agreement with one or more airlines to allow customers to accumulate frequent flier miles for every x amount spent on service.
- Free pay-per-view movies - Maybe your financial service offers free pay-per-view movies under certain conditions. Using service credits, you can set up your system to accumulate the free pay-per-view movies and use the free movies to offset actual movies viewed by the customer.
- Any other type of loyalty program where the customer earns credits that may later be redeemed in some way.

To participate in a program such as those described above, the customer is linked to a service credit membership. The membership record provides the following functionality:

- It defines the accounts that are linked to the membership.

Fastpath: Refer to *Who are the Members?* for information about linking persons and accounts to a membership.

- It defines a *membership type*, which controls certain behavior about the membership.
- It may define an external ID if the membership is associated with an external program, such as a frequent flier mile program.
- It may define a *contract* to use for miscellaneous financial transactions that may get created.
- It may define characteristics used to capture miscellaneous information about the membership.
- Over time, service credit events are created for a membership. The events indicate an amount that either adds or subtracts credit units (i.e., points, miles, dollars, etc) for the membership.

Fastpath: Refer to *The Big Picture of Service Credit Membership* for more information about functionality related to a membership.

How Are Service Credits Earned?

Service credits may be monetary or non-monetary rewards for the financial services availed by the customer.

A typical scenario is that the service credits are earned by the customer on the services linked to his account. For example:

- If a customer signs up for a specific financial service, then he is given complementary reward points for signing up for that financial service. Refer to *Service Credits Earned When Starting Service* for more information.
- For a capital credits membership, capital credit allocations are calculated based on the amount spent by the customer for standard service. For capital credits, a background process is used to calculate the allocated amounts. Refer to *Allocating Capital Credit* for more information.

It is also possible to earn service credits irrespective of other service for the membership's accounts. (Again, it depends on the business rules for the program you are offering.) For example, perhaps you offer 500 frequent flier miles for

signing up for service with your company, regardless of the type of service chosen. In addition, assume that no additional miles are earned for ongoing service. In this example, there is no need to link the membership to any contracts.

Fastpath: Refer to *Contracts Contribute to a Membership* for more information.

Each earned service credit amount is linked to the membership via a *service credit event*.

How Are Service Credits Redeemed?

Once service credits have been earned for a membership, how may a customer redeem these credits? The method by which the credits are redeemed depends on the business rules for the program you are offering. Here are some examples of how a service credit may be redeemed:

- For free pay-per-view movies, perhaps your customer's monthly financial service bill is credited for any pay-per-view movies until all free movies are used. Refer to *Service Credits Redeemed Through Billing* for more information.
- For frequent flier miles, the information about the earned miles is exported to the appropriate airline. The miles are actually redeemed by the customer through the airline, not through your company. Refer to *Interface Membership Information to a Third Party* for more information.
- For a capital credit membership, the company periodically (maybe once a year) decides if credits should be redeemed (referred to as "retired") and if so, how much. The company runs a background process to calculate the "retirement" amount. When an amount is retired, the membership balance is reduced by the retirement amount and the amount is transferred to contracts related to the membership. Refer to *Capital Credit Retirement* for more information.

Note: Tracking Membership Balances. If service credits are redeemed via the system, your membership should probably be configured to keep track of a balance. Refer to *Event Amounts May Contribute to a Balance* for more information.

Each service credit amount redeemed through the system is linked to the membership via a *service credit event*.

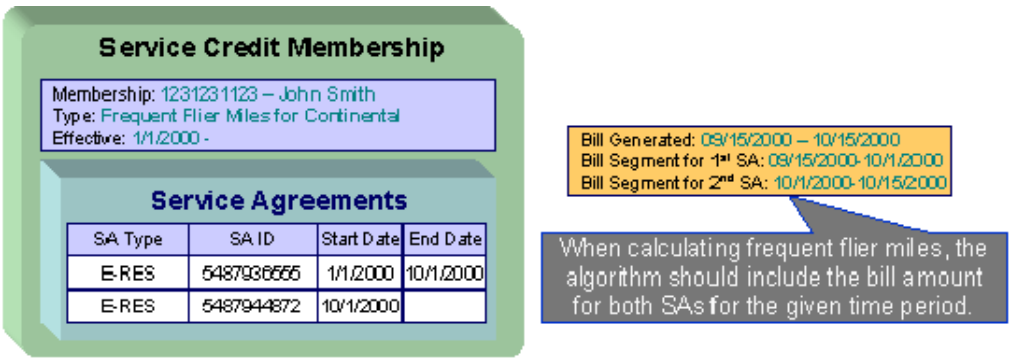
Contracts For a Membership

There are three types of contracts that may be associated with a service credit membership. The following points describe these three types.

Contracts Contribute to the Membership

As described in *how are service credits earned*, many memberships are related to specific contracts for the membership's accounts. We refer to these contracts as the contracts that contribute to the membership because often the service credit amounts earned for the membership are based on amounts spent by the customer for these services.

During the lifetime of a membership, contracts that contribute to a membership may be stopped and other contracts started. For example, perhaps you have a frequent flier membership that is related to financial service. Imagine that the customer starts out with a certain rate for financial service, but later decides to opt for a different type of rate that requires expiring the old contract and creating a new one. How does this affect your frequent flier mile calculation? It depends on how you design the *algorithm* that creates the frequent flier events. Essentially, the algorithm must cater for this situation. The following diagram illustrates the scenario.



The contracts that contribute to a membership are not linked directly to the membership record. This would cause a maintenance burden, requiring links to be updated when service is stopped or started for applicable contracts. Rather, this link is indirect. The list of contract *types* is defined for the membership *type*. The system can determine which contracts are "linked" to the membership by looking at the contract types for the membership type.

Fastpath: Refer to *Determine The Types of Contracts That Contribute to the Membership* for more information on designing your membership type to include appropriate contract types.

Contract Used for Miscellaneous Transactions

For some memberships, you may need to define a special contract to use for miscellaneous transactions. For example:

If events created for a membership cause an adjustment to be created to affect the general ledger, rather than allowing the system to arbitrarily pick a contract to use for this adjustment, the contract to use should be indicated on the membership. Refer to *An Event May Cause Other Actions to Occur* for more information.

Membership Fee Contracts

For some memberships, a *membership fee* may be applicable. A special contract is used to hold the fee. This contract is not linked directly to the membership, but is simply a contract linked to one of the membership's accounts.

Note: Some fees may be refundable. The refunding of a fee must be handled by an algorithm. Refer to *SAST-RF* for information about the algorithm type provided with the base product.

Designing Your Service Credit Options

This section helps you to determine how to design your service credit membership types and service credit event types.

Designing Your Membership Types

This section discusses the options to consider when designing your service credit membership types.

First consider the type of unit that your membership's service credit events represent.

- Is it related to a currency? If so, ensure that your currency is correctly defined on the *currency* page.
- Is it a non-currency unit, such as movies, points or miles? If so, you need to define the unit on the *credit unit* page.

Next, consider other behavior that your membership may exhibit.

- Should your membership *calculate an overall balance*?
- Will miscellaneous financial transactions be created for you membership over its lifetime? If so you may need to link a *contract to your membership*.

- Should events linked to your membership reference a *fiscal year*?

The table below illustrates three types of memberships: one for capital credits and one for frequent flier miles.

Membership Type	Description	Unit Type / Currency or Credit Unit	Has Balance?	Require Contract?	Fiscal Year?
STDCAPCR	Standard capital credit membership	`Currency`/ $	Yes	Yes	Yes
FFDELTA	Delta frequent flier miles	`Credit Unit`/ Miles	No	No	No

The following points explain the settings for each membership above:

- The capital credit membership uses a currency of US dollars for its units. Over time, capital credits are allocated and retired. The overall balance of the credits and debits should be calculated and displayed for information purposes. It is common for the allocation and retirement of capital credits to affect the GL, and as a result, a membership contract is required for posting these financial effects. Finally, in a capital credit situation, the allocation is typically related to a specific fiscal year. When calculating and displaying balances, the balance for each fiscal year must also be available. As a result, the fiscal year must be set to required.
- For the frequent flier membership type, a separate membership type must be created for each different airline. This is because a separate membership must exist for each separate airline in order to keep track of the accumulated miles correctly. The membership type does not have a balance because the accumulated miles are interfaced to the airline and the airline keeps track of the balance. In this example, no contract is required because an assumption is being made that the creation of frequent flier miles does not affect the GL. (Accumulating miles is no liability or expense for the company. The miles are simply accumulated on behalf of a third party.) Finally, the frequent flier miles do not need to indicate a fiscal year.
- For the free pay-per-view movies membership, we assume that the credits are redeemed from within the system. For example, perhaps a rate component calculation algorithm or SQ rule redeems the free pay-per-view movies over time as customers are billed for the movies. As a result, the membership should have a balance. In this example, no contract is required because an assumption is being made that any affect on the GL is posted at the time the free movies are redeemed (i.e., when calculating a bill). (This is just an example. It's possible that you may want to post to the GL when free movies are accumulated to mark a payable for the company.) Finally, free pay-per-view movies do not need to indicate a fiscal year.

More options must be considered for each membership type.

Consider Whether the Membership Should Indicate Subcategories

For some types of memberships, the amount of each service credit event is further grouped by a subcategory. Refer to *Events May Indicate a Subcategory* for more information.

Use subcategories if multiple subtypes of credits may be accumulated and redeemed and if balances need to be tracked for each subcategory.

For our sample membership types, only the capital credits type should indicate subcategories.

Membership Type	Description	Subcategories
STDCAPCR	Standard capital credit membership	Distribution Transportation
FFDELTA	Delta frequent flier miles	

Determine the Types of Contracts 'Linked' to the Membership

Memberships typically exist to reward customers for participating in standard service with the company. Refer to *How Are Service Credits Earned?* for more information. If memberships of this type are related to standard service for your company, determine the contract types for these contracts.

Note: Standard Service Contract Types. These contract types are probably different from the type of contract you may link as the membership contract. The membership contract is used for miscellaneous charges like general ledger posting. These contract types are related to financial service, etc.

For our three examples, assume that the company provides financial service. The capital credits are related only to financial service, frequent flier miles are accumulated for a combination of services.

Membership Type	Description	Contract Types
STDCAPCR	Standard capital credit membership	Financial Services Related
FFDELTA	Delta frequent flier miles	Financial Services Related

Consider Special Functionality Needed When Adding, Activating or Inactivating a Membership

Should a letter or bill message be generated when a membership is created or activated? Should anything happen when a membership is inactivated? If so, you may need to define one or more algorithms to be plugged in on the membership type.

For our examples, let's say that a bill message is generated when a frequent flier or free pay-per-view membership is created. Let's say that a letter is generated when a capital credits membership is activated. We also assume that when a capital credits membership becomes inactive, any outstanding balance is redeemed. For the frequent flier miles and pay-per-view memberships, we assume nothing special occurs when the membership becomes inactive.

Fastpath: Refer to *Lifecycle of a Membership* for more information about the various status values for a membership.

Membership Type	Description	Algorithm System Event	Algorithm
STDCAPCR	Standard capital credit membership	`Membership Activation`	Send Letter
		`Membership Inactivation`	Redeem Balances
FFDELTA	Delta frequent flier miles	`Membership Creation`	Generate Bill Message

Note: Sample Algorithms. The base package does not provide sample algorithms for all the above examples. Refer to *Service Credit Membership Type - Algorithm* for more informationabout the algorithms provided with the system.

Determine Whether The Membership Requires Additional Information

Is there any information about your membership that you need to capture that is not already provided by the base system logic? If the answer is yes, you may need to define characteristics for your membership. Use the *characteristic collection* on the membership type to define the types of characteristics allowed for memberships of this type. You may also define default values for you membership's characteristics.

Designing Your Service Credit Event Types

Now that you have designed your membership types, you need to design the types of service credit events that may be created for your membership.

In many cases, a credit event should cause additional functionality to occur. Algorithms are executed when an event is completed and when an event is canceled and are used to perform additional functionality. The following points illustrate possible algorithms that may be needed when a credit event is completed.

- Validate the event as compared to other events. For example, perhaps a new event should never cause the overall membership balance to fall below zero. You could use an algorithm on the service credit event type to check this condition.
- Create an adjustment that posts to the general ledger. Your credit event may not affect the customer's balance when created, but perhaps it should have an effect on the general ledger. For these types of credit events, you may need to create an adjustment to post to the GL.
- Create adjustments to affect the customer's balance. When "redeeming" a credit, you may need to transfer a monetary amount to one or more of the customer's contracts.
- Create a bill message. Perhaps when a credit is accumulated, you want to inform the customer via a message on the bill indicating the credit amount. A temporary bill message is added to one of the membership's accounts.

Note: One Account Message of the Same Message Type. The temporary bill message collection on the account allows only one bill message of the same bill message type. If it's possible for multiple types of events to be generated for the same account, consider creating a different bill message type for each event type.

- Stamp a batch code and batch run number onto the event. This would be used when your event information needs to be interfaced to an external system.

The following points illustrate possible algorithms that may be needed when a credit event is canceled.

- Validate the event as compared to other events. For example, perhaps canceling an event should never cause the overall membership balance to fall below zero.
- Cancel adjustments that may have been created when the event was completed.

To illustrate examples of when to use some of the algorithms above, we'll design event types for the membership types designed above.

Designing Capital Credit Event Types

The following event types illustrate typical events for a capital credits membership.

SC Event Type	Description	Membership Type	Algorithm System Event	Algorithm
ALLOCATECCR	Capital credit allocation	STDCAPCR	`Event Creation`	Create Simple Adjustment
			`Event Creation`	Generate Bill Message
			`Event Cancellation`	Validate Balance Not < Zero
			`Event Cancellation`	Cancel Related Adjustments

SC Event Type	Description	Membership Type	Algorithm System Event	Algorithm
RETIRECCR	Retire capital credit (apply to customer's balance)	STDCAPCR	`Event Creation`	Validate Balance Not < Zero
			`Event Creation`	Create Adjustments to Affect Customer's Balance
			`Event Cancellation`	Cancel Related Adjustments
FORFEITCCR	Forfeit capital credit (retirement not applied to customer's balance)	STDCAPCR	`Event Creation`	Validate Balance Not < Zero
			`Event Creation`	Create Simple Adjustment (affect GL only)
			`Event Cancellation`	Validate Balance Not < Zero
			`Event Cancellation`	Cancel Related Adjustments

These event types assume the following:

- When credit events are allocated, the customer is notified via a bill message, the general ledger is affected so a simple GL only adjustment is created. If an event of this type is canceled, any adjustments that were created should be canceled. The event amount should always be positive, so checking that the membership balance does not fall below zero is only checked for event cancellation.
- When capital credits are retired, it's possible that the full membership balance is not applied to the customer's balance. For the portion of the retirement that does affect the customer's balance, you need an algorithm that applies the credits to the customer's contracts via adjustments. Cancellation of this event should cause any related adjustments to be canceled. It's assumed that the amount of this event is a credit so checking that the membership balance does not fall below zero is only checked for event completion.
- For the portion of the retirement that is not applied to the customer's balance, the event amount should simply affect the GL so a GL only adjustment is created. Cancellation of this event should cause any related adjustments to be canceled. It's assumed that the amount of this event is a credit so checking that the membership balance does not fall below zero is only checked for event completion.

Note: Service credit event types are independent of subcategories. A capital credits membership typically uses subcategories. When events are created for different subcategories, the same service credit event type may be used. As a result, all subcategories use the same event completion and event cancellation algorithms.

Designing Frequent Flier Event Types

The following event types illustrate typical events for a frequent flier membership:

SC Event Type	Description	Membership Type	Algorithm System Event	Algorithm
ADDMILES	Add miles to the membership	FFDELTA	`Event Creation`	Generate Bill Message
			`Event Creation`	Populate Batch Information
			`Event Cancellation`	Event may not be canceled.

This event type assume the following:

- A new event should generate a bill message.
- Information about the event amount should be interfaced to an external system so batch information should be populated when the event is completed.
- Events of this type may not be canceled because the information is interfaced to an external system. Rather, to reverse an event, simply create a new event whose amount is a credit. This credit amount is also interfaced to an external system.

Note: Sample Algorithm. The system does not provide a sample cancellation algorithm that prevents the event from being canceled.

- There is no validation to ensure that the balance does not fall below zero. Recall that this membership was defined as not requiring a balance.

Notice that only one type of event has been defined for this membership. That is because the credits for this membership are not redeemed via this system. Rather they are accumulated on behalf of an external system.

How Are Service Credit Events Created?

Now that you have designed the behavior of your service credit events, an important issue is to determine how these events are created. Consider your business practice for each type of membership.

Let's use our sample memberships to work through different ways that you may create service credits.

Service Credits for Capital Credit Memberships

For a capital credit membership, capital credit allocations are calculated once a year based on billing history and the cost of service. To accomplish this, a background process calculates the amount and creates the service credits. Refer to *allocating capital credits* for a sample process provided with the base package.

Credits are redeemed via the retirement process. The company determines when to retire capital credits based on analysis of their financial situation. This retirement process is also handled by a background process. Refer to *capital credit retirement* for a process provided with the base package.

For capital credits memberships, special functionality is required when a member dies. The capital credits are considered part of the person's estate and may need to be retired and applied to a beneficiary's account. This process depends on the company's business practice. However, typically, the membership status would change to inactive so that new capital credit allocations are not created for the membership. The system provides a sample inactivation algorithm called *SCMI-RB* that transfers part or all of the outstanding credit balance to the member's contracts. From there, a user can cut a check to the beneficiary.

Note: Some Credits Are Never Retired. For many cooperatives, some types of allocated credits are never retired. Refer to *Partial Retirement* for more information.

Service Credits for Frequent Flier Memberships

In our example, frequent flier miles were related to financial service. Let's assume that frequent flier miles are accumulated every $x spent on financial service. In this example, a bill completion algorithm creates service credits based on the bill segment amounts for these contracts. Refer to *Service Credits Earned Through Billing* for more information.

As mentioned before, the frequent flier miles are not redeemed using this system, but are interfaced to a third party for redemption.

Service Credits for Free Movies Memberships

Note: Sample Algorithms. No base package algorithm types are provided to support the logic described in this example.

In this scenario, let's suppose that a customer receives three free pay-per-view movies when signing up for new service. To handle this, perhaps an contract creation algorithm creates the membership and the service credits when the service is started. Or perhaps you want to wait until the first bill is generated and a bill completion algorithm is used to generate the first credit. It depends on your business practice.

For redeeming the free pay-per-view movies, it is assumed that the movies are credited during billing after it is determined that the customer has been billed for a movie. The number of movies used for the membership is reduced until all the free movies are used.

Fastpath: Refer to *Service Credits Redeemed Through Billing* for more information.

Setting Up Service Credit Options

Setting Up Credit Units

Credit unit is used for service credit membership types whose *events* record non-monetary units. Open **Admin Menu, Credit Unit** to set up credit units.

Description of Page

The following fields display for each credit unit:

Credit Unit The unique identifier of the credit unit.

Description The description of the unit. This also acts as a label for the unit when displaying information about a service credit event.

Symbol / Label Position Indicates whether or not the label for this credit unit appears as a `Prefix` or as a `Suffix` to the service credit event amount.

Decimal Positions Indicates the number of decimal positions used for this credit unit. This information should be used by any algorithm or background process that creates a service credit event to determine how to store the event amount. It is also use to correctly display the service credit amounts.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CR_UNIT*.

Setting Up SC Membership Inactive Reasons

The service credit membership inactive reason must be specified when the status of a service credit membership changes to `inactive`. Open **Admin Menu, SC Membership Inactive Reason** to set up service credit inactive reasons.

Description of Page

The following fields display for each inactive reason:

Inactive Reason The unique identifier of the service credit membership inactive reason.

Description The description of the inactive reason.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SCM_NCTV_RSN*.

Setting Up Service Credit Membership Types

SC Membership Type - Main

Service credit memberships have a membership type. Open **Admin Menu, Service Credit Membership Type** to define the membership type.

Description of Page

Enter a unique **Service Credit Membership Type** and **Description** for each membership type.

Use the **Contract Requirement** flag to indicate whether a *miscellaneous contract* must be linked to memberships of this type. The possible values are `Contract Required` and `Contract Not Allowed`.

Note: Default Note. The value defaults to `Contract Required`.

Use the **Fiscal Year Requirement** flag to indicate whether events linked to memberships of this type should indicate a *fiscal year*. The possible values are `Fiscal Year Required` and `Fiscal Year Not Allowed`.

Note: Default Note. The value defaults to `Fiscal Year Not Allowed`.

Use the **SCM Event Balance** flag to indicate whether or not a *balance of service credit events* linked to memberships of this type should be calculated and displayed. The possible values are `Has a Balance` and `No Balance`.

Note: Default Note. The value defaults to `Has a Balance`.

Use the **SC Membership Type Unit** flag to indicate whether or not the unit for the event amounts for events linked to memberships of this type is `Currency` or `Credit Unit`. When the unit is currency, indicate the **Currency Code**. When the unit is credit unit, indicate the **Credit Unit**.

If events linked to memberships of this type must indicate a subcategory, enter a valid **Subcategory Name** and **Description** for each subcategory applicable to this membership.

Fastpath: Refer to *Events May Indicate a Subcategory* for more information.

Use the **Contract Types** grid to indicate the types of *contracts that contribute to memberships* of this type. Indicate the **Division** and **Contract Type** for each type of contract that is related to a membership of this type.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SCM_TYPE*.

SC Membership Type - Algorithm

Open **Admin Menu, Service Credit Membership Type** and navigate to the **Algorithm** tab to define any algorithms that are associated with a membership type.

Description of Page

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

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

The following table describes each **System Event** for which you can define algorithms.

System Event	Optional / Required	Description
`Membership Creation`	Optional	Use this system event for algorithms that are executed when a membership is created. Click *here* to see the algorithm types available for this system event.
`Membership Activation`	Optional	Use this system event for algorithms that are executed when a membership becomes `active`. Click *here* to see the algorithm types available for this system event. **Note: Creating Memberships In Active Status**. A membership may be created in either `pending` status or in `active` status, based on your business practice. For memberships created in `active` status, the system executes the membership creation algorithms and the membership activation algorithms.
`Membership Inactivation`	Optional	Use a system event of `Membership Inactivation` for algorithms that are executed when a membership becomes `inactive`. Click *here* to see the algorithm types available for this system event.

SC Membership Type - Characteristics

To define characteristics that can be defined for service credit memberships of a given type, open **Admin Menu, Service Credit Membership Type** and navigate to the **Characteristics** tab.

Description of Page

Use the characteristics collection to define characteristics that can be defined for service credit memberships of a given type. Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on service credit memberships of a given type. Turn on the **Default** switch to

default the **Characteristic Type** when service credit memberships of the given type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on.

Setting Up Service Credit Event Types

Service credit events created for a service credit membership have an event type. Open **Admin Menu, Service Credit Event Type** to set up service credit event types.

Description of Page

Enter a unique **Service Credit Event Type** and **Description** for each event type.

Indicate the **Service Credit Membership Type** to which this event type belongs.

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

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

The following table describes each **System Event** for which you can define algorithms.

System Event	Optional / Required	Description
`Event Completion`	Optional	Use a system event of `Event Completion` for algorithms that are executed when an event is completed. Refer to *An Event May Cause Other Actions to Occur* for more information. Click *here* to see the algorithm types available for this system event.
`Event Cancellation`	Optional	Use this system event for algorithms that are executed when an event is `canceled`. Click *here* to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SC_EVT_TYPE*.

Service Credit Examples

In this section, we provide examples of how to define your control tables to support functionality related to different types of service credit memberships. While your company may not define your environment exactly the same way, this section should help solidify your understanding of how to set up your company's service credit options.

Defining Control Tables for a Refundable Fee

If your membership requires payment of a *fee* that is refundable, you must define an contract type to use for the refundable fee.

Adjustment Type for Refundable Fee

This example assumes that the fee is set up similarly to a deposit. The adjustment used to charge the fee would affect the current balance, but not the payoff balance or the general ledger because the fee is not considered a "receivable", rather it is an amount collected and held for the customer.

Create an *adjustment type* for levying the fee. Indicate the fee amount and indicate the adjustment FT creation algorithm *ADJT-CA*, which affects the current balance, but not the payoff balance or the general ledger.

Note: Adjustment Type Profiles. Be sure to add this adjustment type to an appropriate *adjustment type profile* and ensure this profile is linked to your contract type.

Contract Type for Refundable Fee

Create an contract type to use for the fee. This contract type should be marked as not billable.

Indicate the appropriate adjustment type profile that includes the adjustment type to levy the fee.

Indicate an appropriate payment segment type that references the *PSEG-NM* payment segment FT algorithm. This algorithm affects the payoff amount and current amount by the payment amount, which should cause the current amount to become zero and the payoff amount to become a credit for the fee amount when the fee is paid.

Algorithm for Refunding the Fee

The base product provides an algorithm type *SAST-RF* that refunds a fee when *contracts that contribute to the membership* are stopped. You must create an algorithm for this algorithm type and enter the contract type created above as an input parameter.

This contract stop algorithm must be plugged in on all contract types that you defined for the *membership type*.

Defining Control Tables for a Nonrefundable Fee

If your membership requires payment of a *fee* that is not refundable, you can set this up in two ways:

- You can create a special contract type to handle charging the fee. The adjustment contains your fee. You may use this option if the membership is related to multiple types of services and not all services need to be present in order to create the membership.
- You can levy an adjustment on one of the other contracts that is being started. You may use this option if the membership is related to a single service (for example, free pay-per-view movies).

For our example, we will set up the contract for the nonrefundable fee with a separate 'fee' contract type.

Adjustment Type for Nonrefundable Fee

Create an *adjustment type* for levying the fee. Indicate the fee amount and indicate the adjustment FT creation algorithm *ADJT-NM*, which affects the current balance and payoff balance by the adjustment amount, and affects the general ledger.

Note: Adjustment Type Profiles. Be sure to add this adjustment type to an appropriate *adjustment type profile* and ensure this profile is linked to your contract type.

Contract Type for Nonrefundable Fee

Create an contract type to use for the fee. This contract type should be marked as not billable.

Indicate the appropriate adjustment type profile that includes the adjustment type to levy the fee.

Indicate an appropriate payment segment type that references the *PSEG-NM* payment segment FT algorithm. This algorithm affects the payoff amount and current amount by the payment amount, which causes the current amount to become zero and the payoff amount to become a credit for the fee amount when the fee is paid.

Defining an Contract Type for Miscellaneous Transactions

Does your membership require a *contract* to support miscellaneous transactions? If so, you need to consider the contract type to use for this contract. This contract type should be marked as not billable.

If you choose to use the order transaction to set up the membership, this contract type must be defined on the *algorithm that creates the membership*.

Using Campaigns/Packages to Set Up Membership

If enrollment in a membership is a common occurrence for your customers when starting service, you should consider using the *order* page to start service for the customer and to create the membership as well.

Note: This section only makes sense if you are familiar with the *Sales and Marketing* chapter.

The recommendation is to use a question/miscellaneous field to ask the customer service representative to indicate the appropriate membership type. Algorithms validate this membership type and use it to create a membership of that type.

This section walks you through how to set up the campaign / package required to support this.

Column Reference for Membership Type

In order to ask the customer service representative to define an appropriate service credit membership type, you must define a column reference for the membership type.

Add a new column reference with the following information:

- Column Reference: SCM-TYPE
- Description: SC Membership Type
- FK Reference: `SCM TYPE`
- Long Description: Service Credit membership type to use when creating a service credit membership at start time.

Note: Column Reference Algorithms. This column reference indicates a validation algorithm and a posting algorithm. However, we have not defined them yet so for now simply save this information.

Algorithms to Create Membership via Order

The base product provides two algorithm types to support the creation of a membership record via the order page: a validation algorithm type and a posting algorithm type.

Validate Membership Information

This algorithm is a column reference validation algorithm that checks that an input membership type is valid. Refer to *CRVL-ME* for more information. You must define an appropriate algorithm for this algorithm type, and on that algorithm you must define the column reference used to identify the membership type. For example:

- Algorithm: VAL MEM TYPE
- Description: Validate SC Membership Type
- Algorithm Type: `CRVL-ME`
- Parameter1: (Column Reference for Membership Type): SCM-TYPE

Post Membership Information

This algorithm is a column reference posting algorithm that creates a membership using the membership type indicated by the user. Refer to *CRPS-ME* for more information. You must define an appropriate algorithm for this algorithm type, and on that algorithm you must define the column reference used to identify the membership type. For example:

- Algorithm: CREATE MEMBRSHP

- Description: Validate SC Membership Type
- Algorithm Type: `CRPS-ME`
- Parameter1: (Column Reference for Membership Type): SCM-TYPE
- Parameter2: (Service Credit Membership Status): `10`-Pending or `20`-Active. Refer to *Lifecycle of a Membership* for more information.
- Parameters3-4: (Division, Contract Type): Indicate the information needed to create an contract for miscellaneous transactions.

Note: This algorithm first looks for an existing contract of this division / contract type linked to the membership's accounts. If one exists, it uses that contract to link to the membership. If an contract does not exist, it creates one with the input division and contract type.

Note: Update the Column Reference. Now that you have defined the validation and posting algorithms, return to your *column reference for membership type* and define the algorithms there.

Although your company may support multiple types of memberships, this column reference and its algorithms have been designed such that only one column reference for membership type would be needed to set up any type of membership. Although the posting algorithm for the membership type column reference indicates an contract type to use for miscellaneous transactions, a contract is only created and linked if your membership type indicates that an contract is required. As a result, you may use the same column reference for both memberships that require an contract and those that don't require an contract. However, if you have different membership types that require an contract and each uses a different contract type for the membership contract, you need to define more than one posting algorithm and, as a result, more than one column reference.

Define a Campaign for Creating a Membership When Starting Service

Many factors must be considered when *designing your campaigns and packages*. The possible creation of a membership when using the order page is simply another factor to consider.

If you plan to define a question/miscellaneous field to capture a service credit membership type, the available packages linked to the campaign should be ones that are related to memberships. For example, if your membership is related to financial service, it doesn't make sense to create a membership for a campaign designed to generate a one-time charge.

When defining a question/miscellaneous field, you must indicate its applicability. Consider whether a membership type is required, optional or only applicable for certain packages. This helps ensure that your users capture this information when appropriate.

Including the Membership Fee

You must determine the best way to setup your campaign/package to handle the levying of your membership fee, if applicable. Consider some of the following questions.

- If the membership has a fee, is it a refundable fee or a non-refundable fee?
- Is the fee always applicable for the membership? Or is it waived under certain conditions? For example, maybe the fee is applicable if a customer signs up for a single service, but the fee is waived if the customer signs up for two or more services.

The SCM type column reference and question/miscellaneous field are set up to ask the user what type of membership to create. Because the applicability of the fee may differ for each membership type, you should carefully consider the campaign / package setup to levy the fee correctly.

- If a fee is always applicable for a membership, you may consider creating a membership creation algorithm that creates a fee contract to levy the fee when the membership is created.

Note: Sample Algorithm. The base product does not provide a membership creation algorithm to do this.

- If the fee is not always applicable for a membership, you must determine when the fee is applicable. The recommendation is to include the fee contract type in the contracts-to-create collection for the appropriate package. For example,
  - If the fee is applicable when the customer signs up for a single service, you should define a package for each single service that includes one contract to create for the single service and the fee contract as another contract to create.
  - If the fee is waived when the customer signs up for two or more services, you should define a package for each combination of the multiple services. These packages do not include the fee contract in the contracts to create.
  - If the customer signing up for service is a former customer who has returned, perhaps the fee was paid earlier when the customer originally had service. In this case, maybe your business practice is to waive the fee at this time. To do this, you should set up a question/miscellaneous field for the user to mark that this is a returning customer. Based on the answer to this question, perhaps only packages that do not levy any fee are eligible for selection.

Defining Another Person for Your Account

It is common for a capital credit membership to define more than one person for the account being turned on and linked to a membership. A typical example is a married couple. Both spouses are indicated on the account and financially responsible persons and as a result, both are considered *members*.

Column reference algorithms have been provided by the base product to enable linking a second person to your account via the *order* page.

This section walks you through how to set up the column reference and campaign to support this.

Note: Linking A Second Person to the Account. This logic is not restricted to service credit functionality. Any campaign may be designed to include the ability to link a second person to the account being started.

Column References for Additional Person

In order to ask the customer service representative to link an additional person for the account, you must define several column references to use as questions/miscellaneous fields.

- We should assume that the person may already exist in the database. As a result, a question to record the person ID is needed.
- If the person does not already exist, the user should capture the person's name, an ID type and an ID number. Questions for these three fields are needed. The system uses this information to create a new person.
- Whether we are using an existing person or creating a new one, the person's link to the account must include an account relationship type. A question to record the account relationship type is needed.

Add a new column reference for person ID:

- Column Reference: PERSON-ID
- Description: Person ID
- FK Reference: PER
- Long Description: Person ID of the additional person to link to the order's account.

Add a new column reference for person name:

- Column Reference: PERSON-NAME
- Description: Person Name
- FK Reference: not applicable
- Long Description: Name of a new person to create.

Add a new column reference for identifier type:

- Column Reference: PER-ID-TYPE
- Description: Identifier Type
- FK Reference: `ID TYPE`
- Long Description: Identifier type for the primary ID of the additional person to link to the order's account.

Add a new column reference for identifier number:

- Column Reference: PER-ID-NUM
- Description: Person ID Number
- FK Reference: not applicable
- Long Description: Identifier number for the primary ID of the additional person to link to the order's account.

Add a new column reference for account relationship type:

- Column Reference: ACCT-REL-TYPE
- Description: Account Relationship Type
- FK Reference: `ACCT REL`
- Long Description: Relationship type to use for the link between the additional person and the order's account.

Note: Column Reference Algorithms. One of the column references above must indicate a validation algorithm and a posting algorithm. However, we have not defined them yet so for now simply save this information. We recommend using the account relationship type record because that is used for all additional persons.

Algorithms to Link Additional Person via Order

The base product has provided two algorithm types to support the linking of an additional person to the account via the order page: a validation algorithm type and a posting algorithm type.

Validate Addition Person Information

This algorithm is a column reference validation algorithm which checks that either a person id or person name, a valid ID type and ID number are provided and that a valid account relationship type is provided. Refer to *CRVL-PE* for more information. You must define an appropriate algorithm for this algorithm type and on that algorithm you must define the column reference used for the five fields required for this validation. For example:

- Algorithm: VAL ADD PER
- Description: Validate Additional Person Info
- Algorithm Type: `CRVL-PE`
- Parameter1: (Column Reference for Person ID): PERSON-ID
- Parameter2: (Column Reference for Person Name): PERSON-NAME
- Parameter3: (Column Reference for Person ID Type): PER-ID-TYPE
- Parameter4: (Column Reference for Person ID Number): PER-ID-NUM
- Parameter5: (Column Reference for Account Relationship Type): ACCT-REL-TYPE

Post Addition Person

This algorithm is a column reference posting algorithm that may link an existing person to the order's account or create a new person and link that person to the order's account. Refer to *CRPS-PE* for more information. You must define an appropriate algorithm for this algorithm type, and on that algorithm you must define the column reference used for the five fields required for this logic. For example:

- Algorithm: LINK ADDNTL PER

- Description: Link additional person to order's account
- Algorithm Type: `CRPS-PE`
- Parameter1: (Column Reference for Person ID): PERSON-ID
- Parameter2: (Column Reference for Person Name): PERSON-NAME
- Parameter3: (Column Reference for Person ID Type): PER-ID-TYPE
- Parameter4: (Column Reference for Person ID Number): PER-ID-NUM
- Parameter5: (Column Reference for Account Relationship Type): ACCT-REL-TYPE

Note: Update the Column Reference. Now that you have defined the validation and posting algorithms, return to your *column reference for account relationship type* and define the algorithms there.

Design a Campaign to Include Linking an Additional Person

Any campaign related to a specific account may include the questions and miscellaneous fields defined here to create/link an additional person to the account.

Simply create entries in the questions and miscellaneous fields collection to prompt the user to ask for the required information. A question should exist for each column reference created above. The following table illustrates a possible setup.

Seq	Description	Prompt on Order	Column Reference	Dependency
10	Account relationship type for additional person.	If you would like to link another person to this account, please enter an account relationship type.	`ACCT-REL-TYPE`	Must have account
20	Person ID for additional person	If the additional person already exists in the database, enter the person id.	`PERSON-ID`	Must have account
30	Person name for additional person	If you would like to create a new person, please enter the person name.	`PERSON-NAME`	Must have account
40	ID type for additional person	Please enter the ID type of the new person.	`PER-ID-TYPE`	Must have account
50	ID Number for additional person	Please enter the ID number of the additional person.	`PER-ID-NUM`	Must have account

Service Credits Earned When Starting Service

For some memberships, you may want to add service credits when starting the program. For example, the customer gets three free pay-per-view movies when signing on for financial service.

There are various ways that you can accomplish this. You should work with your implementation team to consider the various options to determine the method that best suits your business practice.

- You could use an contract activation algorithm to create a membership and an event for the initial points. Use this method if the points are related to a single type of contract and the points are earned automatically when starting service (i.e., without any human decision to be made).
- You could use a membership creation or membership activation algorithm to generate a service credit event automatically. This method assumes that the decision to create the membership has been made and that the free initial points are always earned for this type of membership (i.e., regardless of the type of service created).
- You could use questions and miscellaneous fields for a campaign/package to determine a customer's eligibility for participation in the membership and for the initial free points. A column reference posting algorithm could create the membership and/or the service credit event for the free points based on the answers to the questions.
- Perhaps the initial free points are only earned after the first bill is generated. Use a bill completion algorithm to generate the initial points.

You may think of other plug-in spots that could be used to generate free initial points based on your business needs.

Service Credits Earned Through Billing

For some memberships, you may accumulate points as a result of billed amounts for other services.

To accomplish this, you must design an algorithm to be executed at billing time. There are various plug-in spots executed at billing time that you may use, but the recommended plug-in is a post completion algorithm on the customer class. This plug-in is executed after all bill segments are frozen and most of the completion logic has occurred.

Your algorithm should determine the *contracts that contribute to the membership* and calculate the service credit amount for those contracts bill segments.

This algorithm should also consider what to do when bill segments that contributed to the event are canceled. The algorithm provide with the base product *CBCM-SC* checks to see if any canceled bill segments are referenced on any previous events. If so, it includes the canceled amount on the new event. This may cause the new event to be a negative amount. The assumption is that over time, earned credits will compensate for the negative event amount. For a membership that *interfaces information to a third party*, it is assumed that the negative event amount is also interfaced.

Service Credits Redeemed Through Billing

For some memberships, your customer may redeem their earned points by receiving a discount on their bill. For example, if your customer has earned one free pay-per-view movie, you can give them a credit the next time they get billed for a pay-per-view movie.

The base product has not provided any algorithms to credit a bill based on earned service credits. This section will identify considerations your implementers should follow when designing algorithms to redeem service credits through billing.

The recommendation is to credit the customer's bill by generating an adjustment using a pre-bill completion algorithm. This algorithm's responsibilities are as follows:

- Determine if the appropriate bill segment(s) contain the charges that need to be credited. For example, if the service credit is for a free pay-per-view movie, determine if the customer has been billed for a pay-per-view movie.
- Determine if the customer's current service credit balance for this program. For example, how many free pay-per-view movie credits are left?
- Create an adjustment to credit the appropriate contract by the eligible credit amount.
- Update the service credit membership balance by creating a new SC event with a credit for the number of points redeemed. Link the adjustment to the event as a `contributed to FT`.
- Consider cancel/rebill situations. If a cancel/rebill has occurred, determine if there is a change to the redeemed credits.

You may wonder why we don't recommend crediting the customer's bill while generating the bill segment. For example, use an SQ rule to determine if any points should be redeemed and use a rate component to generate a bill calculation line with the credit amount.

The reason for this is that cancel/rebill logic is not straightforward. Algorithms executed during rate application should NOT perform any updates, such as updating the service credit membership balance. The balance should be updated using a bill segment freeze algorithm or a bill completion algorithm.

When a cancellation occurs, the service credit balance should be updated to reinstate the redeemed points. Again, this should occur when the cancellation is frozen or at bill completion time. If you perform a cancel/rebill, the calculation of the rebill segment does not have the up-to-date information about the service credit balance because the reinstatement of the points by the canceled segment has not occurred yet.

Designing Your Rate Options for Capital Credits

The capital credit allocation background process relies on certain data configuration in order to function correctly. This section identifies the required data setup.

Fastpath: Refer to *Allocating Capital Credits* for more information.

Identifying SQ and Sales Information for Historical Bill Segments

To allocate capital credits, the background process retrieves billing history for each contract that contributes to the membership for the given fiscal year. The process needs to calculate the service quantity (SQ) amount billed and the sales amount billed for the contract in that year.

Note: Sales Amount. The sales amount refers to the monetary amount billed. For a capital credit allocation, this amount would generally exclude taxes and may exclude other line item amounts from the bill.

In order to calculate the amounts correctly, the background process must determine which bill calculation lines for each bill segment contain the SQ and/or sales information. Characteristics on the bill calculation lines identify which bill lines should be used.

The rest of this section uses examples to illustrate how you may configure your rate options to support this.

Define Characteristics for SQ and Sales

Char Type for identifying bill lines that hold **SQ information**

- Char Type: CCA-SQ
- Description: Capital credit allocation usage
- Type of Char: Pre-defined
- Values: Y
- Char Entities: Rate Component, Bill Calculation Line

Char Type for identifying bill lines that hold **sales information**

- Char Type: CCA-SALES
- Description: Capital credit allocation sales
- Type of Char: Pre-defined
- Values: Y
- Char Entities: Rate Component, Bill Calculation Line

Define Rate Components

Identify each rate schedule that may be linked to a contract that contributes to a capital credit membership.

For each of these rate schedules, identify the rate components whose resulting bill calculation line will contain a billable service quantity that should be included in the SQ calculation for allocating capital credits. For each rate component that qualifies, define the `CCA-SQ` char type (defined above) and a char value of `Y`.

For each rate schedule, identify the rate components whose resulting bill calculation line amount should be included in the sales calculation for allocating capital credits. For each rate component that qualifies, define the `CCA-SALES` char type (defined above) and a char value of `Y`.

Note: Characteristic Information. The system automatically copies characteristic info from a rate component to its resulting bill calculation line if the char type entities include both rate component and bill calc line.

Define Batch Control Parameters

The background process to allocate capital credits *CPCRALOC* receives the char type and char value to identify the bill calculation lines that contain the SQ and sales amounts. Once you have your characteristics defined, update your batch control to include these values as default parameter values.

Designing Bill Factors for Credit Allocation

The *capital credit allocation* process uses an allocation factor in its calculation. A typical capital credit membership may define multiple subcategories, meaning that allocation amounts are calculated each year for the multiple subcategories. The calculation is the same for each subcategory, but the allocation factor differs.

The process has been designed to calculate the allocation for a single subcategory. If your organization requires allocations calculated for multiple subcategories, the process must be run for each subcategory. The allocation process receives a bill factor as an input parameter. As a result, a different bill factor should be set up to define the allocation factor for each subcategory.

For each subcategory, the allocation factor may differ further for the type of customer. The allocation background process expects the bill factor for each subcategory to define a characteristic type of revenue class. The process determines each contract's revenue class by looking at the value defined on its contract type.

Following is an example of bill factors set up for a capital credit membership with two subcategories: transportation and generation.

Characteristic Type for Allocation Bill Factor

Char Type: REV-CLASS

Description: Revenue Class

Type of Char: Pre-defined

Values: (define all the valid revenue class values)

Char Entity: Bill Factor

Bill Factor for Transportation

Bill Factor Id: CCAF-TRANS

Description: Transportation Allocation Factor

Char Type: `REV-CLASS`

Char Source: `Characteristic Collection`

Char Values: (for each year, the new transportation allocation factor for each revenue class must be defined)

Bill Factor for Generation

Bill Factor Id: CCAF-GEN

Description: Generation Allocation Factor

Char Type: `REV-CLASS`

Char Source: `Characteristic Collection`

Char Values: (for each year, the new generation allocation factor for each revenue class must be defined)

Note: The characteristic source is characteristic collection. It is the responsibility of the background process to determine the contract's revenue class and to pass this value into the bill factor routines to retrieve the correct bill factor value.

Note: Estimating Allocation Factors. Often the company needs to estimate the allocation factors for the new fiscal year and may adjust the values several times until the calculated allocation amounts are satisfactory. Refer to *Allocating Capital Credit* for more information.

Partial Retirement

In the cooperative business, it is common to never retire certain capital credit allocation amounts. The amounts that do not retire should be assigned their own subcategory.

When executing the retirement background process, the subcategory to retire may be input to the process. If you have certain subcategories that you do not retire, you would simply run the background process for the subcategories that do retire.

Cooperatives typically retire amounts and transfer the amounts to a beneficiary when a member dies. This is known as "estate retirement". Refer to *Service Credits for Capital Credit Memberships* for more information. If your business practice designates that certain subcategories of allocated amounts do not get retired, this probably holds true for estate retirement as well. If that is the case, your membership inactivation algorithm should be designed to only retire the appropriate amounts by subcategory.

Interface Membership Information to a Third Party

For some memberships, you may accumulate points for a third party, for example accumulating frequent flier miles for an airline. For these types of memberships, you must interface the event information to the third party.

To interface information to a third party, you may choose one of the following options:

- Design an extract program to interface the information
- Use workflow and notification to interface the information via the XAI tool

Interface Via an Extract Program

The service credit event may indicate a batch code and batch run number. Design a program to extract event information to a third party. This extract program would select service credit events marked with its batch code and the current run number.

Your service credit event must define a completion algorithm that stamps the appropriate batch code and run number. The base product provides an algorithm type to perform this logic. Refer to *SCEC-BT* for more information.

Use Workflow & Notification to Interface Info

Workflow and notification allows you to send information to a third party via an extract program or via XAI. The service credit event has its own logic for interfacing via an extract program, so you would use workflow and notification only if you need to interface to the third party via XAI.

Other Considerations For Interfacing Info to a Third Party

Because event information is extracted to a third party, you must consider how to handle adjustments to the event amount. For example, if your event is generated based on the creation of a bill segment, what should happen if that bill segment is cancelled?

You may want to prevent these types of events from getting canceled. Validation like this may be added via a user exit or using an event cancellation algorithm.

Note: Sample Algorithms. The product does not provide any base algorithms to prevent an event from being canceled.

You should allow negative event amounts to be created so that this information may also be sent to the third party's system.

Chapter 10

Defining Loan Options

Topics:

The topics in this section describe how to set up the system to enable loan functionality.

Note: Loans are optional. The system configuration requirements described in this section are only relevant if your organization loans money to customers.

The Terms Of A Loan Are Stored On A Contract

Loans are initiated by creating a loan contract type for a customer. The loan contract type (and its contract type) contains the loan's terms:

- The **loan amount** is held in the contract type's Total Amount to Bill.
- The **customer's periodic payment amount** is held in the contract type's Recurring Charge Amount.
- The **number of amortization periods** (e.g., 36 months, 240 months, etc.) is held in the contract type's Number of Payment Periods.
- If the **interest rate** is the same for all customers with this type of loan contract type, the interest rate is defined on the contract type's contract type (using a bill factor). If a specific interest rate applies to the customer, the contract type's interest rate is defined by specifying a bill factor value on the customer's contract type (where the bill factor value contains the specific interest rate for the customer).
- The contract type controls the **periodicity of the bills** (e.g., monthly or bi-weekly).

Because a loan is defined using a contract type, the typical functionality that is controlled by the contract type's contract type is supported, including:

- How and when it is billed.
- How payments are booked in the GL (and the payment priority relative to other contract types).
- How its debt is monitored by credit and collections.
- How late payment charges are calculated.

Loan contract types are created using *start / stop* just like all other contract types. The start/stop transaction has special loan functionality that allows an operator to specify the contract type-specific loan terms described above.

Note: Automatic calculation of periodic payment amount / number of periods. The system calculates a loan's periodic payment amount or number of payments (whichever is left blank). You can have the system do this on *Start/Stop Maintenance* (using the **Calculate** button that appears on the *start service confirmation window*), and on *Loan - Main* (by clicking the **Calculate** button). Regardless of where you do this, the calculation is performed by an algorithm on the loan's contract type. Refer to the *LPDA-SI* algorithm type for more information about the base package algorithm.

Payoff Balance and Current Balance for Loans

As described under *Current Amount versus Payoff Amount* , a loan contract's current balance and payoff balance differ during the lifetime of the loan. Current balance contains how much the customer owes to remain current (typically their periodic payment amount), and payoff balance contains the amount the customer would have to pay to payoff the loan (typically the principal balance plus any accrued interest charges).

Unlike other contracts, loans have two accounts receivable distribution codes: long term and short term. These two codes allow the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). The current balance for a loan is always the amount of the short-term receivables. The payoff balance for a loan is always the net of the short-term receivables and the long-term receivables.

If the contract has a *special role* of `Loan`, the financial transaction algorithms supplied with the base package transfer the current amount between the long-term receivables and the short-term receivables in the GL. For example, when a bill segment is generated for the loan contract, the amount of the periodic payment is transferred from the long-term receivables to the short-term receivables. Don't worry, the examples in the following sections show exactly what these transactions look like.

An operator can see the how the bills, payments and adjustments have affected the GL, current balance and payoff balance using *Contract Financial History* .

The following sections provide examples of how adjustments, bills and payments are recorded in the GL and the subsequent effect on the current and payoff balances. When reading the examples, remember that the payoff balance is always the net of the short-term receivable and the long-term receivable balances.

Booking The Principal Amount Using An Adjustment

When a loan contract is activated (i.e., when its status changes from `pending start` to `active`), an adjustment is created to book the principal amount. If the customer takes out a loan of 10,000, the adjustment's financial transaction looks as follows:

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan contract is activated (and an adjustment is created to book the principal)	Long Term Loan Receivable 10,000 Cash <10,000>	0	10,000	0	10,000

This adjustment is issued if:

- The contract's contract type indicates a *special role* of `Loan`.
- The loan contract's Total Amount to Bill contains an amount (i.e., the loan amount).
- The adjustment type has been set up as follows:
  - The adjustment type's distribution code should reference the GL account to credit (e.g., Cash).
  - The adjustment type's FT algorithm reference `Payoff Amt = Adj Amt / Current Amt = 0` (booking principal only impacts a customer's payoff balance).

Note that because this financial transaction doesn't have a current amount (the customer doesn't actually owe a current amount yet), there is no need to book anything to the short-term receivables distribution code.

Loan Amortization Schedule Calculation

The amortization schedule is a projection of the amount of principal and interest in each payment over the life of the loan. The amortization schedule may change, for example if the interest rate changes or the customer makes an overpayment (reducing the principal balance).

A loan's amortization schedule is calculated when an operator clicks the **Calculate** button on *Loan - Main*. Please be aware that when this button is clicked, an algorithm plugged in on the loan *contract type* actually calculates the amortization schedule (refer to the algorithm type *LSCH-SI* for more information).

Billing For Loans And Interest Calculation

A bill segment is produced for a loan when its contract's account is next billed.

Factors on a loan's contract type controls when a bill segment is produced for a loan. Typically contract types for loan contracts are set up to use anniversary *calendar billing* . For this configuration:

- You must indicate the type of anniversary billing in the calendar billing option. Currently, we only support the `Anniversary Future Billing` (meaning that loans are billed in advance just like a classic home loan

is). Refer to *Using The Anniversary Method* for more information about how this billing method controls the end date of the loan bill segment.

- You must reference *an anniversary billing frequency* consistent with the *recurring charge frequency* (e.g., monthly, quarterly, etc.).

To set up for a loan that is billed on a monthly basis, you would define the following fields in Contract Type - Billing:

- **Use Calendar Billing**: Anniversary Future Billing
- **Anniversary Bill Frequency**: Monthly
- **Total Bill Amount**: Required
- **Recurring Charge**: Required
- **Recurring Charge Frequency**: Monthly
- **Total Amount To Bill Label**: Loan Amount
- **Recurring Chg Amt Label**: Payment Amount

If your type of loan must be billed on an exact date (for example, always on the 15 th of the month) or with an exact number of days between each bill (for example, every 14 days), then your loan should be set up to use the calendar billing option of `Use Bill Period`. In order for your loan bills to be created on specific dates, the system makes the following assumptions:

- Your *bill cycle schedule* for the loan's account is defined with the dates that you want the loan to bill and is defined with the window start date equal to the window end date.
- You define a *bill period* schedule corresponding to your bill cycle schedule. Each bill period schedule's bill date should match your bill cycle window start date and each bill period schedule's bill segment end date should be set to the loan period end date.
- Considerations for the first bill. If the loan contract type's Initial Start Date Option indicates that the first day of the contract should be billed, then the loan contract's start date should match the window start date of the next bill cycle schedule for the account. If the loan contract type's Initial Start Date Option indicates that first day of the contract should not be included, then the loan contract's start date should be one day prior to the window start date of the next bill cycle schedule for the account.
- Define your *recurring charge frequency* to match the frequency of your bill periods.

How the bill segment affects the customer's balance, and how it affects the general ledger are controlled by several algorithms defined on the loan contract's *contract type* and *bill segment type* :

- The contract type's loan schedule algorithm controls how the *loan amortization schedule* is calculated.
- The contract type's loan interest charge algorithm controls how interest is calculated.
- The bill segment type's create algorithm controls how the bill lines are constructed.
- The bill segment type's financial algorithm controls how the general ledger is affected by the bill.

The 2 nd entry in the following table contains an example of the financial transaction produced by the base package algorithms (note, the first *financial* transaction in the table was described under *Booking The Principal Amount Using An Adjustment*).

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan contract is activated	Long Term Loan Receivable 10000 Cash <10000>	0	10,000	0	10,000 long-term: 10,000
First bill segment is produced	Interest: Long Term Loan Receivable 41.66 Interest Revenue <41.66> Transfer Long Term To Short Term: Short Term Loan Receivable 438.71 Long Term Loan Receivable <438.71>	438.71	41.66	438.71	10,041.66 short-term: 438.71 long-term: 9,602.95

Several important points are illustrated above:

- When a bill segment is produced for a loan contract, the following takes place (if you use a bill segment creation algorithm of `Create Bill Segment for Loans`):
  - The loan contract type's *bill segment creation* algorithm calls the contract type's *loan interest charge* algorithm.
  - The base package *loan interest charge* algorithm calculates simple interest based on: 1) unbilled principal (i.e., the contract's payoff balance minus the current balance), 2) the number of billing periods covered by the bill, and 3) the *interest rate* . Refer to the algorithm type *LINT-SI* for more information about the base package interest calculation algorithm.

Note: No interest on interest and no interest on past due amounts. Just like a classic home loan, the base package algorithms do not calculate interest on interest, nor do they calculate interest on past due amounts. If you want to levy a late payment charge, use the contract type's late payment processing. If your organization calculates interest differently, you must develop your own algorithm(s).

- The contract type's *bill segment creation* algorithm uses the calculated interest to format the bill segment's bill lines. It creates one bill line to show the amount of interest in the payment, and another bill line to show the amount of principal. The principal amount is equal to the contract's periodic payment amount minus the amount of calculated interest.

- The financial transaction illustrated above is created if you use a bill segment financial algorithm of `Payoff Amt = Interest / Current Amt = Bill Amount` on the loan's bill segment type. The following explains how this algorithm works:
  - The contract's *current balance* increases by the amount of the loan's periodic payment amount (i.e., its recurring charge amount). In other words, the amount the customer thinks they owe increases by 438.71.
  - The contract's *payoff balance* increases by the amount of interest. In other words, if the customer wanted to payoff the loan, they'd owe 10,041.66.
  - The *Interest* portion of the GL accounting is straightforward (if you're an accountant). It simply takes the amount of interest and debits it to the contract type's receivable distribution code (long-term receivables) and credits it to the distribution code defined on the interest rate's bill factor (typically interest revenue).
  - The *Transfer* portion of the GL accounting transfers moneys from long-term receivables (i.e., the unbilled principal) to short term receivables (the billed portion of the debt). The amount transferred is equal to the FT's effect on the contract's current balance, allowing the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). Remember that the payoff balance is the net of the short-term and long-term receivables.

Paying What Is Owed

When a payment is made for a loan:

- The contract's payoff amount is reduced by the payment amount.
- The contract's current amount is reduced by the payment amount.

The events that happen when a customer makes a payment against a loan is controlled by the FT algorithm plugged in on the loan contract's payment segment type. We'll use an example to help explain how this algorithm works. The 3 rd entry in the following table illustrates the financial transaction produced when a payment is made (note, the first two financial transactions were described above).

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan contract is activated	Long Term Loan Receivable 10000 Cash <10000>	0	10,000	0	10,000 long-term: 10,000

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
First bill segment is produced	Interest: Long Term Loan Receivable 41.66 Interest Revenue <41.66> Transfer Long Term To Short Term: Short Term Loan Receivable 438.71 Long Term Loan Receivable <438.71>	438.71	41.66	438.71	10,041.66 short-term: 438.71 long-term: 9,602.95
Payment is made	Affect Cash Cash 438.71 Long Term Loan Receivable <438.71> Transfer Long Term To Short Term: Long Term Loan Receivable 438.71 Short Term Loan Receivable <438.71>	-438.71	-438.71	0	9,602.95 short-term: 0 long-term: 9,602.95

The financial transaction illustrated above is created if you use a payment segment FT creation algorithm of `Payoff Amt = Current Amt = Pay Amt` on the loan's payment segment type. The following explains how this algorithm works:

- The contract's *current balance* decreases by the amount of the payment amount. In other words, the customer thinks they owe 0 after the payment. Note refer to *Overpayments* for information about how an overpayment affects the contract's balances and the general ledger.

- The contract's *payoff balance* decreases by the amount of the payment. In other words, if the customer wanted to payoff their loan, they'd owe 9,602.95.
- The *Cash* portion of the GL accounting is straightforward (if you're an accountant). It simply takes the amount of the payment and debits it to the payment segment type's distribution code (typically cash) and credits it to the contract type's receivable distribution code (long-term receivable).
- The *Transfer* portion of the GL accounting effectively reduces short-term receivables by the FT's effect on the customer's current balance. This reduction is handled by an offsetting increase to long-term receivables (to make up for reduction made when the cash was applied). Again, this differentiation between short-term and long-term receivables allows the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term).

Overpayments On Loans

You can determine whether you want to accept loan overpayments. Overpayments reduce the principal amount (the amount owed on the loan), which follows the philosophy adopted by a typical home loan.

When the payment is made, any overpayments are distributed according to the overpayment distribution algorithm defined for the customer class. Any customer classes for which you want to allow loan overpayments should use an overpayment distribution algorithm that keeps the overpayment on the loan contract.

When the payment transaction is frozen, the system checks to see if there is a credit amount on the loan contract's current balance. If a credit exists, the customer has made an overpayment and an adjustment is created to zero out the current balance and transfer the amount of the credit from the contract's short-term receivable to long-term receivable. The adjustment may appear on the customer's next bill to show the additional amount paid against the principal.

The algorithm that controls this adjustment to remove the credit on current balance is plugged in on the loan contract's contract type and is applied on the `Contract Type - Payment Freeze` system event.

The 3 rd and 4 th entries in the following table illustrate an overpayment (note, the first two financial transactions were described above).

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan contract is activated	Long Term Loan Receivable 10000 Cash <10000>	0	10,000	0	10,000 long-term: 10,000

Event	**GL Accounting**	**Effect On Current Balance**	**Effect On Payoff Balance**	**Current Balance**	**Payoff Balance**
First bill segment is produced	Interest: Long Term Loan Receivable 41.66 Interest Revenue <41.66> Transfer Long Term To Short Term: Short Term Loan Receivable 438.71 Long Term Loan Receivable <438.71>	438.71	41.66	438.71	10,041.66 short-term: 438.71 long-term: 9,602.95
Payment is made (with an overpay-ment of 200.00)	Affect Cash Cash 638.71 Long Term Loan Receivable <638.71> Transfer Long Term To Short Term: Long Term Loan Receivable 638.71 Short Term Loan Receivable <638.71>	-638.71	-638.71	-200.00	9,402.95 short-term: <200.00> long-term: 9,602.95

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Create adjustment to remove contract type's credit.	Transfer Short Term Credit to Long Term: Short Term Loan Receivable 200.00 Long Term Loan Receivable <200.00>	200.00	0	0	9,402.95 short-term: 0 long-term: 9,402.95

In the 3 rd financial transaction illustrated above, the billed amount of the payment works essentially the same as that illustrated under *Paying What Is Owed*. If the `Keep Overpayment on Loan Contract` algorithm is plugged in on the overpayment distribution event on the customer class, the overpayment amount is applied to the loan contract type, creating a credit on the contract type's current balance. The following explains how this algorithm works:

- If the account has an loan contract type and there is an excess credit, the credit is applied to the loan contract type (as long as this does not cause the loan contract type to have a credit payoff balance).
- If there is not a loan contract type to which the credit can be applied, the algorithm checks to see if there is an open excess credit contract type for the account.
    - If so, the excess credit amount is applied to the excess credit contract type.
    - If not, the algorithm creates an excess credit contract type and applies the amount to this contract type.

The 4 th financial transaction illustrated above is created if the `Create Adjustment to Remove Contract's Credit algorithm is plugged in on the loan's contract type's payment freeze event.` The following explains how this algorithm works:

- `If the contract's current balance is less than zero, the algorithm creates a frozen adjustment that removes the credit by transferring the credit from the short-term receivable to the long-term receivable.`
- The contract's *current balance* increases by the amount of the credit transfer. In other words, the customer thinks they owe 0 after the transfer.
- The contract's *payoff balance* doesn't change because the payoff balance is always the net of the short-term and long-term receivables. In other words, if the customer wanted to payoff their loan, they'd still owe 9,402.95.

Note: Overpayments and interest. The base package interest calculation algorithm (plugged in on the loan's contract type) does not take into consideration the exact date that the overpayment is made when calculating the interest for the period. It only takes into consideration the outstanding principal amount (payoff balance - current balance) at the time of the interest calculation.

Adjusting Loan Amounts

You would issue ad hoc adjustments if you need to change a loan's payoff and/or current balance outside of the normal billing / payment functions.

An adjustment can:

- Reduce a loan's payoff balance.

- Reduce a loan's current balance (i.e., how much the customer thinks they currently owe).
- Reduce both the loan's payoff and current balance.

For adjustments that affect payoff amount only:

- These adjustments are used to change the principal owed, e.g., if an additional amount is loaned.
- The adjustment's adjustment type should reference the `Payoff Amt = Adj / Current Amt = 0` FT algorithm.
- GL lines will be generated to reflect the change to principal.

For adjustments that reduce current amount only:

- These adjustments can be used to change the amount that the customer must pay as part of the next bill.
- The adjustment's adjustment type should reference the `Payoff Amt = 0 / Current Amt = Adj Amount (no GL)` FT algorithm.
- Typically, GL lines are not generated for FTs that only affect the customer's current balance. However, moneys must be transferred from long to short in the amount of the adjustment (as described above under *Payoff Balance and Current Balance for Loans*).

For adjustments that reduce both payoff and current amount:

- These adjustment types can be used to levy additional charges, such as late fees, or to correct interest calculations.
- The adjustment's adjustment type should reference the `Payoff Amt = Adj / Current Amt = Adj` FT algorithm.
- GL lines are generated to reflect the change to principal. In addition, GL lines must be generated to transfer money from long to short in the amount of the adjustment (as described above under *Payoff Balance and Current Balance for Loans*).

Note: Adjustments that affect the principal balance (payoff balance - current balance) affect the term of the loan because interest is based on the principal balance.

Note: Adjustments can cause credit balances to exist. If you credit the loan contract, it is possible for the current balance to become negative. You may need to create additional adjustments that affect the current amount, depending on whether the customer needs to pay this amount as part of the next payment.

Writing Off Loans

Loans are written off using the standard write-off processing.

Fastpath: Refer to *The Big Picture Of Write Off Processing* for background information.

We illustrate the classic financial transactions that transpire to financially write-off a loan to help illustrate important points (these are the 4 th and 5 th entries in the following table):

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Loan contract is activated	Long Term Loan Receivable 1000 Cash <1000>	0	10,000	0	10,000 long-term: 10,000
First bill segment is produced	Interest: Long Term Loan Receivable 41.66 Interest Revenue <41.66> Transfer Long Term To Short Term: Short Term Loan Receivable 438.71 Long Term Loan Receivable <438.71>	438.71	41.66	438.71	10,041.66 short-term: 438.71 long-term: 9,602.95
Payment is made (with an overpay-ment of 200.00)	Affect Cash Cash 638.71 Long Term Loan Receivable <638.71> Transfer Long Term To Short Term: Long Term Loan Receivable 638.71 Short Term Loan Receivable <638.71>	-638.71	-638.71	-200.00	9,402.95 short-term: <200.00> long-term: 9,602.95

Event	**GL Accounting**	**Effect On Current Balance**	**Effect On Payoff Balance**	**Current Balance**	**Payoff Balance**
Create adjustment to remove contract's credit.	Transfer Short Term Credit to Long Term: Short Term Loan Receivable 200.00 Long Term Loan Receivable <200.00>	200.00	0	0	9,402.95 short-term: 0 long-term: 9,402.95
Sync up current with payoff balance	Transfer Long Term To Short Term: Short Term Loan Receivable 9,402.95 Long Term Loan Receivable <9,402.95>	9,402.95	0	9,402.95	9,402.95 short-term: 9,402.95 long-term: 0

Event	GL Accounting	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Transfer balance to a write-off contract	Transfer From Loan Contract: Write Off Xfer Clearing 9,402.95 Long Term Loan Receivable <9,402.95> Long Term Loan Receivable 9,402.95 Short Term Loan Receivable <9,402.95> Transfer To Write Off Contract: Bad Loans Expense 9,402.95 Write Off Xfer Clearing <9,402.95> Note, these will cause a balance of 9,402.55 to exist on the write-off contract.	-9,402.95	-9,402.95	0	0

The only unusual portion of the last two financial transactions is the impact on short and long term receivables. Please see the examples above under *Billing For Loans And Interest Calculation* and *Paying What Is Owed* to understand how any impact to a loan's current balance causes this type of financial transfer to occur.

Distribution Codes for Loans

As explained above under *Payoff Balance and Current Balance for Loans*, loans have two accounts receivable distribution codes: long term and short term. These two codes allow the general ledger to differentiate between unbilled loan receivables (long term) and billed loan receivables (short term). Both receivables distribution codes are defined on the loan contract type.

In addition, loans have a distribution code used to book interest revenue. The interest revenue distribution code is defined on the loan's bill factor value for a revenue class (defined on the loan's contract type). For example, on the bill factor you can use one distribution code to book interest revenue from the financial services revenue class and another distribution code to book interest revenue from the other financial services revenue class. In this example, you create two loan contract types, one for financial services revenue and the other for commercial revenue.

Loans may also have a bad loan debt (expense) distribution code that is used when writing off a loan. The bad loan debt distribution code is defined on the loan's write-off contract type. Refer to *Defining Credit & Collections Options* for more information.

Setting Up The System To Enable Loans

The above topics provided background information about how loans are supported in the system. The following discussion summarizes the various setup tasks alluded to above.

Distribution Code

You must set up the following *distribution codes* :

- Long term receivables
- Short term receivables
- Interest revenue
- Bad loan debt

Adjustment Types

The following adjustment types are needed:

- Activate a loan. This should reference a distribution code associated with cash or the cash equivalent and an FT algorithm of `Payoff Amt = Adj / Current Amt = 0`.

Note: Creating Checks for Loan Amounts. If you want the system to initiate a check to the customer for the loan amount, the loan activation adjustment type should indicate an **A/P Request Type Code**. Refer to *Controls The Interface To A/P & 1099 Reporting* for more information.

- Remove Credit (Overpayment). This adjustment should reference an FT algorithm of `Payoff Amt = 0 / Current Amt = Adj Amount (no GL)`. Even though this algorithm indicates that there is no effect on the GL, there is a special exception built in for loan contracts. The special exception creates GL details to transfer the current balance (short-term receivable) to the long-term receivable. Refer to *Overpayments* for more information.

Note: Printing Overpayments on Bills. If you want the overpayment adjustment to appear on customers' bills, turn on **Print by Default** and enter a **Description on Bill** (e.g. "Additional Principal"). For more information, refer to *Controls Information Printed On The Bill*.

- Adjustment types to perform any of the adjustments described under *Adjusting Loan Amounts* .

Adjustment Type Profile

Create an adjustment type profile that references the adjustment types used on a loan. Besides the above adjustment types, it should also reference adjustment types to levy late payment charges (if applicable), levy non-sufficient funds charges (if applicable), refund overpayments (if applicable), sync current balance with payoff balance at write-off time, transfer balances to a write-off contract, and write-down small balances.

Algorithms

Add the following *algorithms*:

- Overpayment Distribution. This algorithm is later plugged in on the customer class for the `Overpayment Distribution` system event. Refer to the algorithm type *OVPY-LO-CSA* for more information about the base package algorithm.
- Bill Segment Creation for Loans. This algorithm is later plugged in on the loan's bill segment type. Refer to the algorithm type *BSBS-LO* for more information about the base package algorithm.
- Bill Segment Financial Transaction Creation for Loans. This algorithm is later plugged in on the loan's bill segment type. Refer to the algorithm type *BSBF-LO* for more information about the base package algorithm.
- Amortization Schedule. This algorithm is later plugged in on the contract type for the `Loan Schedule` system event. Refer to the algorithm type *LSCH-SI* for more information about the base package algorithm.
- Interest Calculation. This algorithm is later plugged in on the contract type for the `Loan Interest Charge` system event. Refer to the *LINT-SI* algorithm type for more information about the base package algorithm.
- Payment Periods/Payment Amount Calculation. This algorithm is later plugged in on the contract type for the `Loan Periods and Amount` system event. Refer to the algorithm type *LPDA-SI* for more information about the base package algorithm.
- Loan Contract Payment Freeze. This algorithm has a parameter that must reference the Remove Credit adjustment type defined above. This algorithm is later plugged in on the contract type for the `Payment Freeze` system event. Refer to thealgorithm type *STPZ-RMVCR* for more information about the base package algorithm.

Bill Factor

You must set up a *bill factor* that defines the interest rate. If you have different interest rates for different types of loans, you can create a separate bill factor for each interest rate.

On the bill factor's *bill factor value* , make sure to reference the GL distribution code used to book interest revenue for the revenue class specified on the loan's contract type.

Customer Class

Any *customer class* on which you want to allow overpayments for a loan must use the overpayment distribution algorithm defined above. The overpayment distribution algorithm keeps the overpayment on the loan contract rather than transferring it to an excess credit contract, allowing a subsequent adjustment to apply the overpayment to the principal balance.

Bill Segment Type

Create a bill segment type that references the bill segment creation algorithm defined above and the FT creation algorithm defined above.

Frequency

Create *frequency* codes to correspond to the frequency of your loans. When setting up your contract types, you must indicate a recurring charge frequency and, if you use the `Anniversary Billing Option`, you must indicate an anniversary frequency.

Bill Period

If you use the `Use Bill Period` option, set up a bill period with an appropriate schedule of dates for billing your loan.

Bill Cycle Schedule

If you use the `Use Bill Period` option, the bill cycle schedule for these types of loans should be defined with an appropriate schedule of dates for billing your loan.

Collection and Write Off Processes

You should set up the appropriate credit and collections information. Refer to *Defining Credit & Collections Options* for more information.

Contract Type

You must set up a *contract type* for your loan contracts (you may need multiple contract types if you have different business rules for different types of loans). The following points describe the minimal requirements for a loan contract type.

Contract Type - Main

Define the following options:

- **Distribution Code** should be a long-term receivable code.
- **Service Type** should references something like "Miscellaneous Service".
- Specify the **Revenue Class** that, together with the interest bill factor, determines the distribution code used to book the loan interest revenue.
- The **Payment Segment Type** should reference the `Normal Payment`. The base package payment segment financial algorithm (`Payoff Amt = Current Amt = Pay Amt )` used for Normal Payment pay segment types creates the additional GL details to transfer the credit from long-term receivables to short-term receivables if the contract's special role is Loan.
- Turn on **Late Payment Charge** if applicable.
- Define an appropriate **Adj** ustment **Type (Synch Current)** that will cause current balance to be synchronized with payoff balance (if the loan is *written off*).

Contract Type - Detail

Define the following options:

- **Special Role** is `Loan`.
- Specify the **Interest Bill Factor** set up above.
- Use **Override Interest Flag** to indicate whether the interest rate defined on the interest bill factor may be overridden at the contract level. If you select `Allowed`, the interest rate may be overridden by a contract value on the contract.
- Use the short-term receivable account defined above as the **Loan A/R Distribution Code**. If you do not want the system to differentiate between short-term receivables and long-term receivables, make the loan A/R distribution code the same as the distribution code (above).

Contract Type - Billing

Fastpath: For an overview of some of these options, be sure to refer to *Billing For Loans And Interest Calculation* for more information.

Define the following options:

- The *Bill Segment Type* should reference the value created above.
- **Characteristic Location Req** uired should not be checked. (A loan contract is not a location-based service. contract types that have this box checked are filtered out of the contract type search when the start method is Start a contract, so users will never be able to start a loan contract that requires a characteristic location.)
- **Use Calendar Billing** must equal `Anniversary Future Billing` or `Use Bill Period`.

- **Bill Period** is required for the `Use Bill Period` option.
- **Anniversary Billing Frequency** is required for the `Anniversary Future Billing` option and must equal the periodicity of the bills (monthly, weekly, etc.).
- **Total Bill Amount** is required (it holds the principal).
- **Total Amount To Bill Label** should reference something like "Loan Amount".
- **Recurring Charge Amount** is required (it holds the periodic payment amount).
- **Recurring Chg Amt Label** should reference something like "Payment Amount".
- **Recurring Charge Frequency** is required (it defines the periodicity associated with the recurring charge amount) and must be the same as the bill period frequency or the anniversary billing frequency (based on your **Use Calendar Billing** option).

Contract Type - Rate

Turn off the **Rate Required** switch as loans do not use rates.

Contract Type - Adjustment Profile

Adjustment Type Profile should reference the profile set up above.

Contract Type - Credit and Collections

The **Debt Class** and **Write Off Debt Class** should reference an appropriate value consistent with your credit and collections rules. Refer to *Defining Credit & Collections Options* for more information.

Contract Type - Algorithms

The `Loan Schedule`, `Loan Interest Charge`, `Loan Periods and Amounts` and `Payment Freeze` *algorithms* defined above must be set up.

Chapter 11

Defining Non-billed Budget Options

Topics:

A non-billed budget (NBB) is a payment plan that allows your customers to pay set amounts at specified intervals (e.g. every two weeks). Non-billed budgets are typically used when your company bills on an infrequent basis and you want to provide your customers with a mechanism to make smaller payments more frequently. As the name suggests, bills are not created for the non-billed budget's scheduled payments; customers must remember to make their payments at the scheduled intervals. The topics in this section describe how to design and set up non-billed budgets.

Note: Non-billed budgets are optional. The system configuration requirements described in this section are only relevant if your organization offers non-billed budgets.

What Is A Non-billed Budget?

A non-billed budget is a special type of budget or payment plan that encompasses three major elements:

- A set of scheduled payments
- The business rules used to recommend and potentially renew the payment schedule
- The business rules that govern the financial impact on the current and payoff balances of the contracts covered by the payment schedule

Non-billed budgets are managed via a contract whose contract type has a special role of `Non-billed Budget`. If a contract type has a special role of non-billed budget it must have one or more recommendation rules, and contracts of that type are allowed to have payment schedules.

The Financial Impact Of Non-billed Budgets

When you set up the non-billed budget contract type, you can indicate whether the non-billed budget is monitored by the *account debt monitor* process. The following sections contain examples of financial transactions for non-billed budgets that are monitored by the account debt monitor.

Note: Unmonitored non-billed budgets. Contracts that are covered by unmonitored non-billed budgets are subject to different financial treatment than those that are monitored. The *financial transactions relevant for unmonitored non-billed budgets* are discussed later.

Current Amount For Contracts Covered By A Non-billed Budget

As described under *Current Amount versus Payoff Amount*, the values of the current balance and payoff balance are the same for most financial transactions. One exception is for contracts that are covered by a non-billed budget in which case the current balance is always zero and the payoff balance is always the amount the customer really owes.

When a non-billed budget is activated or an contract is added to a non-billed budget, the system creates adjustments for all affected contracts to set the current amount equal to zero. The adjustment type that references the `Payoff Amt = 0 / Current Amt = Adj Amount (no GL)` algorithm is taken from the contract's contract type.

Note: Non-billed Budgets and Open Item Accounts. If the non-billed budget is for an open item account, no match event is created for the financial transaction that reduces the current amount to zero. This FT must be manually matched if required.

The following example shows the effect of activating a non-billed budget that covers a financial services contract with a current balance of 25. The transactions for the financial services contract are illustrated on the left and the transactions for the non-billed budget contract are illustrated on the right.

Event	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
	Financial Services Contract				Non-Billed Budget Contract			
Starting balance	0	0	25	25				

Event	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
Nonbilled budget is activated	-25	0	0	25				

By setting the current balance for the covered contracts to zero, the contracts are insulated from the regular debt monitoring process. Depending on the non-billed budget *recommendation algorithm*, the payoff balance can be ignored, divided evenly between the scheduled payments or added to the first scheduled payment.

Activating a non-billed budget has no affect on its own current or payoff balances.

Scheduled And Actual Payments On The Non-billed Budget

When a scheduled payment is due, an adjustment is created to increase the non-billed budget's current balance by the expected amount. The current balance on the contract can be monitored to ensure that payments are being made on time.

When the payment is made, the non-billed budget's current balance is reduced to zero and the non-billed budget's payoff balance reflects the accumulated credit from the payment. This accumulated credit is transferred to the covered contracts when the next bill for the account is completed.

The following example illustrates scheduled and actual payments for the non-billed budget in the previous example. The amount of the scheduled non-billed budget payments is 10. Note that the first two transactions were described above.

Event	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
	Financial Services Contract				Non-billed Budget Contract			
Starting balance	0	0	25	25				
Nonbilled budget is activated	-25	0	0	25				
1st scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10

An algorithm (`Process NBB Scheduled Payment`) plugged in on the non-billed budget contract type creates the appropriate financial transactions when the scheduled payment is due. The algorithm is called by the Non-billed Budget Scheduled Payment Processing (*NBBPS*) background process.

The normal payment processing handles the adjustments created when the payment is made.

Note: Non-billed Budget Payment Cancellation. If a payment is canceled, the financial transaction is reversed.

Note: Automatic Payments And Non-billed Budgets. Users may set up a non-billed budget to use *automatic payments* by setting up the account's auto pay options. Users may also exclude the non-billed budget from automatic payment if the account is set up for automatic payment. The *NBB Scheduled Payment Automatic Payment Create* background process calls the *Auto Pay Creation* algorithm to create the auto payment for a non-billed budget.

Overpayments for Non-billed Budgets

Typically, payments in excess of the non-billed budget's current balance are credited to an overpayment (excess credit) contract. When the next adjustment is created for a scheduled payment, the credit on the overpayment contract is used to relieve the non-billed budget current balance.

Note: Overpayment Distribution Algorithm. The overpayment distribution is a function of the overpayment distribution algorithm plugged in on the account's customer class. We strongly recommend that the non-billed budget contract type be set up so that overpayment is not allowed. Any excess payments should go to an overpayment contract. For more information about overpayment distribution, refer to *Overpayment Segmentation*.

In the example below, the customer pays 20 for a scheduled payment instead of 10. The non-billed budget contract is illustrated on the right and the overpayment contract is illustrated on the left. The first four transactions were illustrated above.

Event	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
	Non-billed Budget Contract				Overpayment Contract			
Starting balance								
Nonbilled budget is activated								
1st scheduled payment is due	10	0	10	0				
Payment	-10	-10	0	-10				
2nd scheduled payment is due	10	0	10	-10				
Overpayment	-10	-10	0	-20	-10	-10	-10	-10
3rd scheduled payment is due	10	0	10	-20				-20
Transfer Adjustment	-10	-10	0	-30	10	10	0	0

After the `Process NBB Scheduled Payment` algorithm creates the next scheduled payment, it looks for a credit amount on the overpayment contract and creates an adjustment to transfer the credit balance (or the amount of the payment if the credit is more than the scheduled payment amount) from the overpayment contract to the non-

billed budget contract. The overpayment transfer adjustment type and the overpayment contract type are specified as parameters to the algorithm.

Underpayments For Non-billed Budgets

An insufficient payment or a canceled payment leaves a current balance on the non-billed budget contract.

In the example below, the customer makes a payment of 7 for the fourth scheduled payment. The example below does not show the overpayment contract (illustrated above). The first eight transactions are discussed above.

Event	**Effect On Current Balance**	**Effect On Payoff Balance**	**Current Balance**	**Payoff Balance**	**Effect On Current Balance**	**Effect On Payoff Balance**	**Current Balance**	**Payoff Balance**
	Financial Services Contract				Non-billed Budget Contract			
Starting balance	0	0	25	25				
Nonbilled budget is activated	-25	0	0	25				
1st scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10
2nd scheduled payment is due					10	0	10	-10
Overpayment					-10	-10	0	-20
3rd scheduled payment is due					10	0	10	-20
Transfer Adjustment					-10	-10	0	-30
4th scheduled payment					10	0	10	-30
Underpayment					-7	-7	3	-37

The `Process NBB Scheduled Payment` algorithm creates a trigger to ensure that the current balance is monitored by the account debt monitor. Refer to *Credit And Collections And Non-billed Budgets* for more information.

Billing For Contracts Covered By The Non-billed Budget

When the next bill for the account is completed, the credit on the non-billed budget is transferred to the covered contracts. The credit is prorated over the covered contracts according to the relative payoff balances on each contract.

In the example below, the financial services contract's bill is 33. The first ten transactions are discussed above.

Event	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance	Effect On Current Balance	Effect On Payoff Balance	Current Balance	Payoff Balance
	Financial Services Contract				Non-billed Budget Contract			
Starting balance	0	0	25	25				
Nonbilled budget is activated	-25	0	0	25				
1st scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10
2nd scheduled payment is due					10	0	10	-10
Overpayment					-10	-10	0	-20
3rd scheduled payment is due					10	0	10	-20
Transfer Adjustment					-10	-10	0	-30
4th scheduled payment					10	0	10	-30
Underpayment					-7	-7	3	-37
Bill	0	33	0	58				
Bill completion (transfer adjustment)	0	-37	0	21	0	37	3	0

When a bill segment financial transaction is created, the current amount is set to zero for contracts that are covered by the non-billed budget (if you use a bill segment FT algorithm of `Payoff Amt = Bill Amt / Current Amt = Amt Due`). Though not evident by the name, the `Payoff Amt = Bill Amt / Current Amt = Amt Due` algorithm does set the current amount to zero for monitored non-billed budgets. (For contracts with other roles, the current amount is equal to the amount due or the recurring charge.)

A bill completion algorithm transfers money from the non-billed budget to the covered contracts (if you plug in the `NBB Credit Transfer` bill completion algorithm on the non-billed budget contract type). The algorithm type supplied with the base package distributes the credit using the method described in *Distributing Non-billed Budget Credit*.

Note: Canceled Bill Segments. No new processing occurs when a bill segment is canceled; any credit balance remains on the covered contract.

Distributing Non-billed Budget Credit

Both the NBB Credit Transfer bill completion algorithm type and the non-billed budget contract stop algorithm type supplied with the base package use the same method of distributing a credit from a non-billed budget contract to the covered contracts. The following points describe how the credit is distributed:

- Covered contracts that are already in credit (due to some other circumstance, such as a cancellation and rebill) are excluded from the distribution.
- The distribution to each covered contract will not exceed its total payoff to ensure that none of the covered contracts have a credit balance.
- The credit is prorated over the covered contracts according to the relative payoff balances on each contract.
- The calculation of the payoff balance is adjusted to exclude the current balance to ensure that the credit is prorated over the debt covered by the budget, not any ad-hoc debt for the contract.

Note: The current balance on covered contracts should always be zero. The only exception occurs if an adjustment has been added that directly affected the contract balance. In this case, the distribution assumes that the balance is outside the non-billed budget and needs to be paid separately.

- Any excess credit remains on the non-billed budget contract until the next distribution takes place or until the *non-billed budget contract is stopped*.
- The type of adjustments created is determined by the **Adjustment Type (Xfer)** specified on the non-billed budget contract type.

The examples in the table below illustrate the points above.

NBB Contract Credit	Contract 1 Payoff	Contract 1 Current	Contract 2 Payoff	Contract 2 Current	Contract 1 Credit Xfer	Contract 2 Credit Xfer
-100.00	-100.00	0.00	-200.00	0.00	0.00	0.00
-100.00	150.00	0.00	-50.00	0.00	-100.00	0.00
-300.00	150.00	0.00	50.00	0.00	-150.00	-50.00
-100.00	150.00	0.00	250.00	0.00	-37.50	-62.50
-100.00	150.00	0.00	250.00	50.00	-42.86	-57.14

The first example above shows that covered contracts with a credit balance are excluded from the distribution. Any excess credit remains on the non-billed budget.

The second example shows that one covered contract has a credit balance, so the entire credit is distributed to the remaining contract.

The third example shows the amount distributed to a covered contract does not exceed its payoff balance. Again, any excess credit remains on the non-billed budget contract.

The fourth example illustrates how the credit is prorated based on the payoff balance. The fifth example illustrates the same prorating but with a current balance on one of the contracts (contract 2). (Remember that the prorating excludes any current balance.)

The prorated amount is calculated by subtracting the current balance from the payoff balance then multiplying the result by the distribution amount and dividing by the total payoff owing of all covered contracts.

Stopping a Contract Covered By a Non-billed Budget

If a service agreement covered by a non-billed budget is stopped, the system must bring the current balance and payoff balance of the covered contract back in synch. The system creates an adjustment using the Synch Current adjustment type from the contract's contract type.

In the example below, the financial services contract's payoff balance is 21. The first twelve transactions are discussed above.

Event	**Effect On Current Balance**	**Effect On Payoff Balance**	**Current Balance**	**Payoff Balance**	**Effect On Current Balance**	**Effect On Payoff Balance**	**Current Balance**	**Payoff Balance**
	Financial Services Contract				Non-billed Budget Contract			
Starting balance	0	0	25	25				
Nonbilled budget is activated	-25	0	0	25				
1st scheduled payment is due					10	0	10	0
Payment					-10	-10	0	-10
2nd scheduled payment is due					10	0	10	-10
Overpayment					-10	-10	0	-20
3rd scheduled payment is due					10	0	10	-20
Transfer Adjustment					-10	-10	0	-30
4th scheduled payment					10	0	10	-30
Underpayment					-7	-7	3	-37
Bill	0	33	0	58				
Bill completion (transfer adjustment)	0	-37	0	21	0	37	3	0
Contract is stopped	21	0	21	21				

Note: In addition to synching the current and payoff balance, the contract being stopped is removed from the covered contracts collection. When the last covered contract is removed from the collection, the non-billed budget is also *stopped*.

Financial Transactions For Unmonitored Non-billed Budgets

Some companies have the concept of non-billed budgets where the payments made by the customer are optional. This functionality is implemented as an unmonitored non-billed budget. Unmonitored non-billed budgets allow a customer to make optional prepayments towards a bill.

As explained previously, unmonitored non-billed budgets receive different financial treatment than monitored non-billed budgets. The financial transaction algorithms use the **Non-billed Budget Monitor** flag on the non-billed budget's contract type to create the appropriate financial transactions. The following table describes the differences in the financial treatment of monitored and unmonitored non-billed budgets.

System Event	Monitored Non-billed Budgets	Unmonitored Non-billed Budgets
When a non-billed budget is activated or an contract is added to a non-billed budget...	The system creates adjustments for all affected contracts to set the current balance equal to zero.	The current balances for covered contracts are not zeroed. The activate contract and non-billed budget maintenance processing do not create any adjustments if the non-billed budget is unmonitored.
When a scheduled payment is due...	An adjustment is created to increase the non-billed budget's current balance by the expected amount.	There is no change to the non-billed budget's current balance; it is always zero. The scheduled payment processing background process does not execute the *NBB process scheduled payment algorithm*.
When the payment is made...	The non-billed budget's current balance is reduced to zero and the non-billed budget's payoff balance reflects the accumulated credit from the payment.	Typically the payments are distributed to an excess credit (overpayment) contract. Refer to *Distributing Payments for Unmonitored Non-billed Budgets* for more information.
When a bill segment financial transaction is created...	The current amount is set to zero for contracts that are covered by the non-billed budget.	`The covered contracts' current amounts are equal to their payoff amounts.`
At bill completion...	Money from the non-billed budget is transferred to the covered contracts. Customers may receive a bill for information purposes, but they are not required to pay it.	Money from the overpayment contract is transferred to the account's contracts. The customer is still liable to pay the outstanding balance.

System Event	Monitored Non-billed Budgets	Unmonitored Non-billed Budgets
When the non-billed budget is stopped or an contract is removed from a non-billed budget...	The system creates adjustments for all affected contracts to synchronize their current balances with their payoff balances, thus removing the zero current balance and replacing it with the actual current balance.	There is no change to the current balances for covered contracts as the balances already reflect the actual current balance.

Unmonitored non-billed budgets also receive different credit and collections treatment. Refer to *Credit and Collections and Unmonitored Non-billed Budgets* for more information.

Distributing Payments for Unmonitored Non-billed Budgets

For non-billed budgets, payments are distributed according to the payment and overpayment algorithms on the customer class. The base package payment distribution algorithm applies a payment first towards any contracts that have overdue or current balances (refer to *Distributing A Payment* for more information). Since the unmonitored non-billed budget contract doesn't have a current balance, it is not considered by the payment distribution algorithm. If there aren't any contracts with a current balance, the overpayment distribution algorithm handles the remaining credit. You can elect to:

- Apply the overpayment to an excess credit contract. This is the method that we strongly recommend because all financial transactions are then a function of the normal payment, overpayment and billing processes.
- Apply the overpayment to the highest priority contract that is eligible for overpayment (as specified on the contract type). You can use this method to apply the overpayment to the unmonitored non-billed budget contract. If you use this method, you must also set up the system to *transfer the credit from the unmonitored non-billed budget*.

Note: Use An Excess Credit contract. We strongly recommend that payments for unmonitored non-billed budgets are distributed to an excess credit contract. In this case, the non-billed budget is just a shell to hold the covered contracts and recommend a payment schedule; all financial transactions are a function of the normal payment, overpayment and billing processes.

Fastpath: Refer to *Overpayment Segmentation* for a detailed discussion of overpayment distribution options.

Transferring Credit from Unmonitored Non-billed Budgets

If you distribute an overpayment to an unmonitored non-billed budget contract (i.e. the unmonitored non-billed budget maintains a credit balance instead of an overpayment contract), you must plug-in a bill completion algorithm on the contract type to transfer the credit balance to the covered contracts at bill completion time. The bill completion algorithm type (*BCMP-NB*) supplied with the base package transfers the credit balance to the covered contracts when the bill is completed. Additionally, the **Adjustment Type (Xfer)** on unmonitored non-billed budget contract types should reference a FT algorithm of `Payoff Amt = Adj / Current Amt = Adj` to ensure that the credit is removed from both the current and payoff balances.

Caution: You must create your own contract stop algorithm type for correctly stopping an unmonitored non-billed budget that maintains a credit balance. The contract stop algorithm that is supplied with the base package does NOT transfer remaining credit from the unmonitored non-billed budget contract. (The base package contract stop algorithm transfers the remaining credit using the overpayment distribution algorithm on the customer type, which you have set up to transfer to back to the unmonitored non-billed budget.)

Designing Non-billed Budgets

The topics in this section describe functionality that you must consider when designing non-billed budgets.

Making Contracts Eligible For Non-billed Budgets

A billable contract may be covered by a non-billed budget when its contract type is flagged as `Eligible for Non-billed Budget`.

All contracts that are eligible for non-billed budgets should reference a bill segment type that uses the `Payoff Amt = Bill Amt / Current Amt = Amt Due bill segment FT algorithm. This algorithm sets the contract's` current amount to zero if it is covered by a monitored non-billed budget.

A list of the contracts covered by a non-billed budget is maintained with the non-billed budget contract. This list is used at bill completion to determine the financial transactions that should occur.

Designing Recommendation Rules

Users (and the renewal process) ask the system to recommend the scheduled payments for a non-billed budget. In general, this recommendation process must establish:

- The amount to be paid
- The dates on which the payments are due

We envision many different types of recommendation rules. For example:

- Recommend 26 scheduled payments to be made on a fortnightly basis that are due on Tuesdays.
- Recommend monthly payments that are due on the nearest workday after the 10th of the month.
- Recommend scheduled payments where the customer pays their annual charges in 10 out of 12 months where the payments are not due in November and December.
- Recommend bi-monthly payments where the payments are due on the first workday following the 5th and 20th of the month.

Additionally, the recommendation rules must determine how to handle any outstanding payoff balances for covered contracts. The true-up rule provided with the base package *payment schedule algorithm type* can ignore the payoff balance, divide the payoff balance evenly between the scheduled payments, or add the payoff balance to the first scheduled payment.

A recommendation rule comprises three elements:

- An algorithm to calculate an average daily amount (created using the *NBDA-DA* algorithm type provided with the base package)
- Two algorithm types are provided with the base package to calculate a schedule of payments. *NBPS-MON* calculates a monthly schedule and *NBPS-PS* calculates a scheduled based on a specified number of days.
- A collection of default parameter values for the payment schedule algorithm type

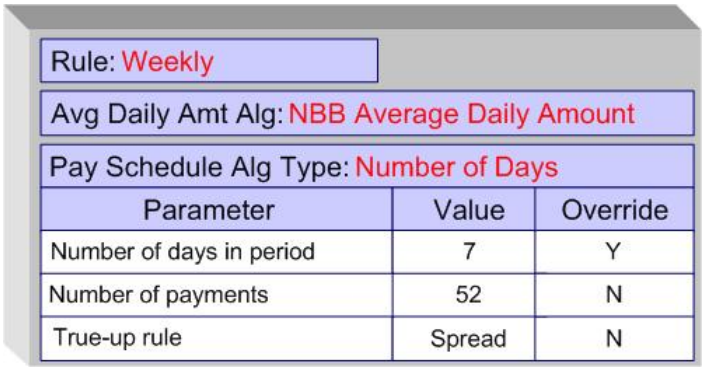


Figure 1: Weekly Payments Recommendation Rule

Note: Additional Parameters. Not all of the parameters associated with the weekly payment schedule algorithm type are illustrated. Refer to the *NBPS-PS* algorithm type for a detail description of the parameters.

The default parameter values for the payment schedule algorithm type may change over time, so the collection contains an effective date. If default values are changed, these changes do not affect non-billed budgets already in effect. Existing non-billed budgets keep the parameter values that were used when the non-billed budget was started.

A user may override the default parameter values for the payment schedule algorithm type to customize the schedule if an override is allowed for a parameter. Additionally, a user may edit the payment schedule details at any time (provided the payment has not yet been processed).

The parameter values used for the recommendation rule are kept with the non-billed budget contract, so that any customized parameter changes can be re-applied to a renewed non-billed budget. For example, the parameter that determines the payment due day may default to the first of the month. To customize the schedule, this value may be changed to the fifth of the month. This amended value is kept with the non-billed budget contract to ensure that the renewed budget follows the same monthly schedule.

Note: Normally parameter values for an algorithm type are kept with the algorithm. Because the parameters may vary for each non-billed budget, the parameter values are kept with the contract in the case of non-billed budgets.

Refer to *Non-billed Budgets Recommendation Rule - Main* for information on creating recommendation rules.

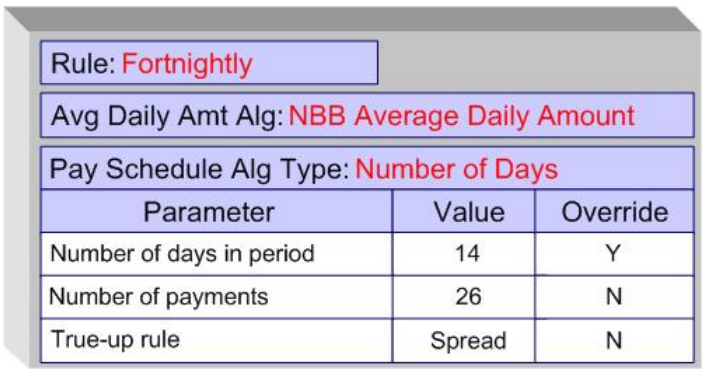
Example Recommendation Rules

The following examples may be helpful in designing and implementing your recommendation rules.

Note: Developing Your Own Payment Schedule Algorithms. The base package comes supplied with a *monthly payment schedule algorithm type*. You can use this algorithm as an example when creating payment schedule algorithm types for your implementation.

Fortnightly Payments Recommendation Rule Example

The following diagram illustrates a recommendation rule where the customers pay every two weeks. The current balance for any covered contracts is added to the first payment.



Rule: Fortnightly

Avg Daily Amt Alg: NBB Average Daily Amount

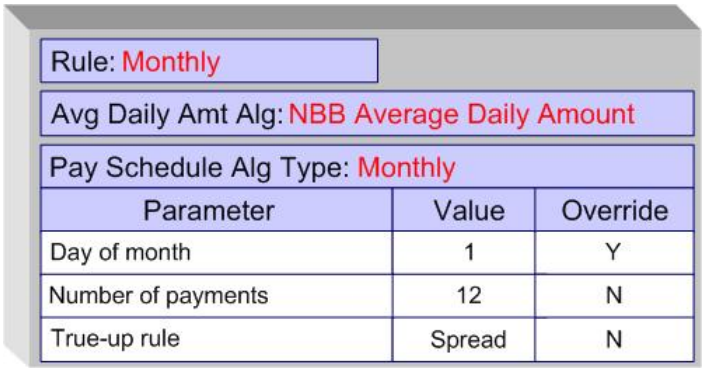
Pay Schedule Alg Type: Number of Days

Parameter	Value	Override
Number of days in period	14	Y
Number of payments	26	N
True-up rule	Spread	N

Figure 2: Fortnightly Payments Recommendation Rule

Monthly Payments Recommendation Rule Example

The following diagram illustrates a recommendation rule where the customers pay twice monthly on the first of the month. The current balance for any covered contracts is spread out over the scheduled payments.



Rule: Monthly

Avg Daily Amt Alg: NBB Average Daily Amount

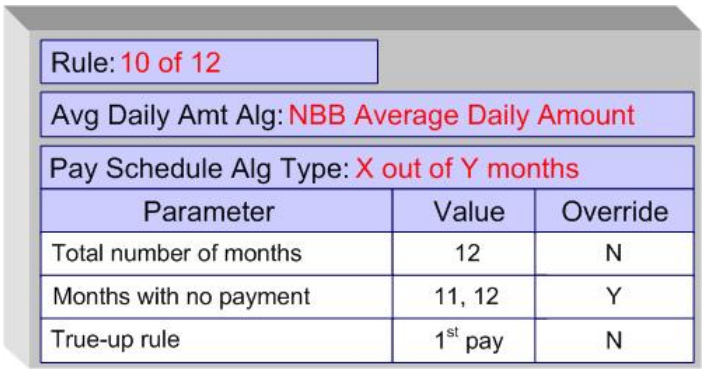
Pay Schedule Alg Type: Monthly

Parameter	Value	Override
Day of month	1	Y
Number of payments	12	N
True-up rule	Spread	N

Figure 3: Monthly Payments Recommendation Rule

Ten Out of Twelve Months Recommendation Rule Example

The following diagram illustrates a recommendation rule where the customers pay one a month except for months during a holiday season (November and December). The current balance for any covered contracts is added to the first payment.



Rule: 10 of 12

Avg Daily Amt Alg: NBB Average Daily Amount

Pay Schedule Alg Type: X out of Y months

Parameter	Value	Override
Total number of months	12	N
Months with no payment	11, 12	Y
True-up rule	1st pay	N

Figure 4: Ten Out of Twelve Months Recommendation Rule

Activating Non-billed Budgets

You can plug in an algorithm (of type *SACR-AT*) on the Contract Creation system event on the non-billed budget contract type to automatically activate the non-billed budget (i.e. transition it from `pending start` to `active` status). If you don't use a Contract Creation algorithm to activate the non-billed budget, it is activated the next time the *Contract activation background process* runs.

When a non-billed budget is activated, you can perform special processing using an algorithm plugged in on the Contract Activation system event on the non-billed budget contract type. The special processing can be developed to do anything that you would like, for example you could:

- Create a customer contact that with an appropriate letter template can generate a letter to inform the customer of their payment amount and payment schedule.
- Initiate the creation of a payment coupon book for a customer.

The system comes supplied with a sample algorithm type (called *SAAT-CC*) that simply creates a customer contact to indicate that the non-billed budget is activated.

Renewing Non-billed Budgets

A non-billed budget can be renewed either manually or via a background process. When the non-billed budget contract is created, the expiration date, renewal date and the recommendation rule used to create the initial budget are kept with the contract. A renewal flag on the non-billed budget contract type controls if a renewal is required, optional or not allowed. If renewal is required, a user must specify a renewal date when creating the contract. The renewal date is defaulted on to an contract based on the valued of the **Days Before Expiration for Renewal** field on the contract type.

An algorithm on the contract type can customize the processing required to renew an contract.

The *contract renewal background process*:

- Executes the contract renewal algorithm (specified on the contract type) when the renewal date is reached (i.e., it is on or before the process date). The base package comes with an algorithm type (*SARN-NB*) that determines the current recommendation rule for a non-billed budget and executes the associated payment schedule algorithm using the non-billed budget contract-specific parameter values to generate a new schedule. It returns new expiration and renewal dates.
- If the renewal algorithm is successful, the renewal and expiration date fields on the contract are updated with the new values.
- If the renewal process is not successful, a To Do list entry (of type *TD-SARN*) is created for the account and contract.

The new payment schedule that is returned from the renewal process for a non-billed budget is appended to the current schedule.

A user can manually launch the renewal process for a non-billed budget contract by clicking the **Renew NBB** button on the *non-billed budget maintenance page*.

Expiring Non-billed Budgets

Non-billed budget contracts may specify an expiration date. The *contract expiration background process* initiates the stop process for all pending start or active contracts where the expiration date is reached (before or on the process date).

Stopping Non-billed Budgets

When a non-billed budget stop is initiated, either on request or because it has expired and is not being renewed, the non-billed budget is transitioned to `pending stop` status. You can plug in an algorithm (of type *SAIS-ST*) on the Contract Stop Initiation system event on the non-billed budget contract type to automatically finalize and stop the contract (i.e. transition it to `stopped` status). If you don't use a Contract Stop Initiation algorithm, the non-billed budget is stopped the next time the *Contract activation background process* runs.

To finalize a pending stop contract, the system first calls the stop contract algorithm plugged-in on the Contract Stop system event on the contract type. The stop contract algorithm type (*SAST-NB*) supplied with the base package:

- Distributes any credit on the non-billed budget to the covered contracts (using the method described in *Distributing Non-billed Budget Credit*)
- Distributes any excess credit remaining on the non-billed budget using the *overpayment distribution* algorithm for the account's customer class and the overpayment transfer adjustment type (specified as a parameter to the algorithm)
- Creates a trigger to cause the account to be reviewed by the account debt monitor
- Creates a customer contact (if the customer contact class and customer contact type parameters are populated)

Caution: Warning! If you do not plug in an contract stop algorithm that transfers the credit balance from the non-billed budget to its covered contracts (or an excess credit contract), the stopped non-billed budget may have a credit balance. You must then manually distribute this credit.

After the contract stop algorithm is finished, the contract stop processing performs the following steps if the contract type has a special role of non-billed budget:

- If the non-billed budget is monitored, create adjustments to synchronize the current and payoff balance of covered contracts using the **Adj. Type (Synch Current)** adjustment type from the covered contracts' contract types
- Remove the covered contracts from the non-billed budget
- Remove all the scheduled payments from the non-billed budget
- Create an adjustment to synchronize current and payoff on the non-billed budget contract using the **Adj. Type (Synch Current)** adjustment type from the non-billed budget's contract type

Note: Synchronizing current and payoff effectively sets the current amount to zero on the non-billed budget contract, as the payoff amount should have been reduced to zero by the distribution and overpayment processing in the algorithm for contract Stop.

Fastpath: Refer to *The Lifecycle Of A Contract* for more information about how a `pending stop` contract is `stopped` and `closed`.

Automatic Payment and Non-billed Budgets

If a customer wants to pay their non-billed budget scheduled payments automatically, the account must be set up for automatic payment. In addition, the non-billed budget must indicate that automatic payment is being used.

Fastpath: Refer to *How To Set Up Automatic Payment For A Non-billed Budget* for more information.

When this is done, a background process referred to as NBBAPAY creates automatic payments on the scheduled payment date by calling the automatic payment creation algorithm plugged in on the installation record.

Note: You must also ensure that your auto pay creation algorithm supports non-billed budget scheduled payments. Note that the *APAY-CREATE* algorithm type supplied with the base package supports non-billed budget scheduled payments.

Credit and Collections and Non-billed Budgets

Unless the non-billed budget is *unmonitored*, the *account debt monitor* (ADM) monitors a non-billed budget contract's current amount just as it does for any other contract. The *scheduled payment algorithm* creates a trigger to ensure that the account debt monitor reviews the account the next time it runs. The review date on the trigger record is set to the process business date.

A separate *debt class* is needed for non-billed budget contract types, thus allowing you to define collection class controls, debt criteria and collection process templates specifically for non-billed budgets. The debt criteria should be set up to trigger a collection process when the arrears amount exceeds $0.01 for more than n payment periods plus the number of grace days that you want to allow.

The collection process template can perform any of the events in standard collection processes, such as sending letters to customers. When the system subsequently stops the non-billed budget, the system removes the covered contracts from the non-billed budget and synchronizes their current balances with their payoff balances. Since the contracts have

current balances again, they are subject to the account debt monitor, which can start subsequent collection processes for any of the contracts that meet the debt criteria for their debt class.

Fastpath: Refer to *Stopping Non-billed Budgets* for a complete description of the events that occur when a non-billed budget is stopped.

Customers can catch up on their payments and avoid having their non-billed budget broken as long as their current balance doesn't violate the debt criteria for the non-billed budget's collection process.

Credit and Collections and Unmonitored Non-billed Budgets

If the **Non-billed Budget Monitor** flag on the non-billed budget's contract type is set to `Unmonitored`, the *NBB Scheduled Payment Processing* background process does not create a trigger for the account debt monitor. Additionally, the non-billed budget's current amount is always equal to zero, so it never violates any debt collection criteria. We recommend using a debt class that has the **Eligible for Collection** flag turned off, such as the N/A debt class.

For unmonitored non-billed budgets, the current balance is kept on the covered contracts so they are subject to the account debt monitor and any debt criteria for their contract types' debt classes. For more information, refer to *Financial Transactions for Unmonitored Non-billed Budgets*.

Non-billed Budget Status

A non-billed budget contract's status is just like any other contract's status. In addition, you can use a characteristic to keep an explicit status relevant to the non-billed budget:

- A base package break non-billed budget algorithm type (*NBBR-BRK*) creates a characteristic value to indicate if a non-billed budget is "canceled" (i.e. manually stopped by a user).

An active non-billed budget implicitly has a "kept" status (i.e. all scheduled payments have been made).

Alerts For Non-billed Budgets

The system provides alerts to highlight the existence of non-billed budgets. These alerts are important to assist the customer service representatives:

- An alert is displayed if the account has a non-billed budget that is not stopped (e.g., `pending start`, `active` or `pending stop`).
- When a user denies a non-billed budget (for whatever reason), the user should create a *customer contact* with a given *customer contact type*. This type of alert prevents the customer from shopping around. An existing alert algorithm type (*CC BY TYPCL*) can highlight these customer contacts.
- For customers who are permanently forbidden from having a non-billed budget, the user should put a permanent *alert on the account*.
- Use an algorithm to highlight cancelled non-billed budgets in the **Alerts** dashboard. The algorithm type to do this is not provided. Use *PP BY STATUS* and *CCAL-WF* as examples of how to create this type of algorithm.

Non-billed Budget Recommendation Rule

Recommendation rules are used to recommend scheduled payments for non-billed budgets. For information about designing recommendation rules, refer to *Designing Recommendation Rules*.

To define recommendation rules, navigate to **Admin Menu, NBB Recommendation Rule.**

Description of Page

Enter an easily recognizable **Recommendation Rule** code and **Description** for each recommendation rule.

Specify the **Average Daily Amount Algorithm** used to calculate the average daily amount for this recommendation rule.

Specify the **Payment Schedule Algorithm Type** used to create the recommended payment scheduled for non-billed budgets that use this recommendation rule. The Payment Schedule Algorithm Type cannot be modified if a non-billed budget contract that is not stopped or cancelled is using this recommendation rule.

Payment Schedule Parameters enables you to define collections of default parameter values for the payment schedule algorithm type that are effective dated. For each collection:

- **Effective Date** defines the date on which the collection of parameter values becomes effective.
- **NBB Rule Parameter Value** specifies the default value of each parameter supplied to the algorithm. Note that the *payment schedule algorithm type* controls the number and type of parameters.
- **Override Flag** indicates whether the user can override the default value for the parameter.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_NB_RULE*.

Setting Up The System To Enable Non-billed Budgets

The above topics provided background information about how non-billed budgets are supported in the system. The topics in this section describe how to set up the system to enable non-billed budget functionality.

Note: Example Setup. This section describes typical non-billed budget configurations. Your set up and configuration may differ depending on your business needs. This section is provided for guidance only. Read the descriptions of non-billed budget functionality above to understand the implications of the described setup.

NBB Distribution Codes

You must set up a Non-billed Budget Clearance *distribution code* that is used on the non-billed budget contract type to credit non-billed payments.

NBB Adjustment Types

Note: Non-billed Budget Financial Transaction Algorithms. The non-billed budget adjustment types use the standard FT algorithm types that are provided with the base package. If you have not yet defined algorithms for these types in your system, do so before creating the non-billed budget adjustment types.

The following adjustment types are needed:

- Add contract To Non-billed Budget. This adjustment type should reference an FT algorithm of `Payoff Amt = 0 / Current Amt = Adj Amount (no GL)`. Because the adjustment affects the current balance only, there is no entry in the GL. This adjustment type is referenced on **Adjustment Type (Current=0)** on contract types that are eligible for non-billed budgets. Note that this adjustment type is never used if the contract is added to an unmonitored non-billed budget.
- Bill Complete for Monitored Non-billed Budget. This adjustment type should reference a distribution code used for balance transfer clearing. It should reference a FT algorithm of `Payoff Amt = Adj / Current Amt = 0`. This adjustment type is referenced on **Adjustment Type (Xfer)** on contract types that are monitored non-billed budgets.
- Synch Balance for Non-billed Budget. This adjustment type should be set up for **Sync. Current Amount** and should reference a FT algorithm of `Payoff Amt = 0 / Current Amt = Adj Amount (no GL)`. This adjustment type is referenced on the **Adj. Type (Synch Current)** on non-billed budget contract types. It is used when a non-billed budget contract is stopped to synch the current balance with the payoff balance.
- Scheduled Payment. This adjustment type should reference an FT algorithm of `Payoff Amt = 0 / Current Amt = Adj Amount (no GL)`. Because the adjustment affects the current balance only, there is no entry in the GL. This adjustment type is referenced on the non-billed budget process scheduled payment algorithm.
- Overpayment Transfer. This adjustment type should reference a transfer distribution code and a FT algorithm of `Payoff Amt = Adj / Current Amt = Adj`. This adjustment is referenced on the non-billed budget process scheduled payment algorithm and the stop non-billed budget algorithm. Note that if there is already a transfer adjustment type created in your system, you do not need to create a new one.

If you are setting up an unmonitored non-billed budget that maintains a credit balance (as opposed to maintaining the credit balance on an overpayment contract), you need to create an adjustment type for Bill Complete for Unmonitored Non-billed Budget. (Refer to *Transferring Credit from Unmonitored Non-billed Budgets* for more information.) The adjustment type should reference a transfer distribution code and a FT algorithm of `Payoff Amt = Adj / Current Amt = Adj`. This adjustment type is referenced on **Adjustment Type (Xfer)** on contract types that are unmonitored non-billed budgets.

NBB Adjustment Type Profiles

Create an adjustment type profile for non-billed budgets that references the following adjustment types:

- Bill complete for monitored non-billed budget
- Bill complete for unmonitored non-billed budget (if used)
- Synch balance for non-billed budget
- Scheduled payment
- Overpayment transfer

Create an adjustment type profile for eligible contracts that references the following adjustment types:

- Add contracts to monitored non-billed budgets
- Bill complete for monitored non-billed budget
- Bill complete for unmonitored non-billed budget (if used)

Note: Bill Complete Adjustment Types. Because the bill complete adjustment types transfer amounts between two contracts, they must be in profiles for both non-billed budget and eligible contract types.

Note: Overpayment Transfer Adjustment Type. The overpayment transfer adjustment type created above is used to transfer funds from an excess credit contract to a non-billed budget when the scheduled payment is processed. It is also used to transfer excess funds from a non-billed budget that is being closed to an excess credit contract. The transfer adjustment should therefore be added to an adjustment type profile that is referenced on the excess credit contract type.

NBB Characteristic Types

Create a non-billed budget status characteristic type that specifies `Contract` as its characteristic entity. The characteristic type should include the following predefined values:

- Non-billed Budget Canceled
- Non-billed Budget Severed

NBB Customer Contact Class And Types

Create a non-billed budget customer contact class. The contact class may include the following customer contact types:

- Non-billed Budget Activate
- Non-billed Budget Renewal
- Non-billed Budget Stop

Note: Customer Contact Letters. If you want to send letters to your customers when a contact of any of these types is created, you must create an appropriate *letter template* and attach it to the contact type.

NBB Algorithms

You must define the following *algorithms*:

- On *NBB recommendation rule*, the Non-billed Budget Daily Amount Calculation. Refer to the *NBDA-DA* algorithm type for more information about the base package algorithm.
- On *Contract type*:
  - Non-billed Budget Process Scheduled Payment. This algorithm has parameters that must reference the Scheduled Payment and Overpayment Transfer adjustment types defined above. Another parameter references an overpayment contract type (that you may need to create if there is not already one in your system). Refer to the *NBPA-PS* algorithm type for more information about the base package algorithm.
  - Non-billed Budget contract Renewal. Refer to the *SARN-NB* algorithm type for more information about the base package algorithm.
  - Break Non-billed Budget contract. Refer to the *NBBR-BRK* algorithm type for more information about the base package algorithm.
  - Contract Activation - automatically activate contract (if you want to automatically activated non-billed budgets when they are created). Refer to the *SACR-AT* algorithm type for more information about the base package algorithm. Note that this same algorithm may be used on many contract types.
  - Contract Activation - create customer contact (if you want the system to create a customer contact when NBB contracts are activated). Refer to the *SAAT-CC* algorithm type for more information about the base package algorithm.
  - Contract Stop - automatically stop contract (if you want to automatically transition non-billed budgets from pending stop to stop when their stop is initiated). Refer to the *SAIS-ST* algorithm type for more information about the base package algorithm. Note that this same algorithm may be used on many contract types.
  - Contract Stop - Stop Non-billed Budget. Refer to the *SAST-NB* algorithm type for more information about the base package algorithm.
  - Bill Completion - Non-billed Budget Credit Transfer. Refer to the *BCMP-NB* algorithm type for more information about the base package algorithm.
- On *bill segment type*, the Bill FT Algorithm (for all contracts that are eligible for NBB). Refer to the *BSBF-BA* algorithm type for more information about the base package algorithm. Note this is the standard bill FT

algorithm type used for common bill transactions. It supports bill FT for non-billed budgets and other contract types. You only need to create it if it doesn't already exist in your system.

Note: Payment Schedule Algorithm Types. For non-billed budget payment schedule algorithm types, you need to define the algorithm types (if you add your own algorithm types). You do not need to define algorithms because the parameter values for the algorithm are defaulted on the recommendation rule and stored with the non-billed budget contract. (Normally the algorithm holds the parameter values.) Refer to the *NBPS-MON* algorithm type for more information about the base package payment schedule algorithm.

NBB Debt Class And Collection Process

Set up a separate debt class, collection class control and collection process template for non-billed budgets according to the information in *Credit and Collections and Non-billed Budgets*.

Contract Types for Contracts Covered by NBBs

You must modify the contract type for any contracts that you want to allow to be covered by a non-billed budget.

- Verify that the specified **Bill Segment Type** (on the Billing page) references a financial transaction algorithm to set the current amount to zero for monitored non-billed budgets. The FT creation algorithm type *BSBF-BA* (i.e., `Payoff Amt = Bill Amt / Current Amt = Amt Due`) supplied with the base package sets the current amount to zero for contracts that are covered by monitored non-billed budgets, though this is not evident by the name. (For contracts that are not covered by a non-billed budget, the current amount is equal to the amount due or the recurring charge.) Refer to *Billing For Contracts Covered By The Non-billed Budget* for more information.
- Set the **Eligible for Non Billed Budget** flag (on the Billing page) to `Eligible for Non-billed Budget`.
- Populate the **Adjustment Type (Current = 0)** (on the Detail page) to indicate the adjustment to be used to zero out the current amount on the covered contracts when the non-billed budget contract is activated. Use the Add Contract To Non-billed Budget *adjustment* created above. This adjustment type is only called for contracts that are covered by a monitored non-billed budget.
- Reference an **Adjustment Type Profile** (on the Adjustment Profile page) that includes the Add Contract To Non-billed Budget adjustment type referenced in the **Adjustment Type (Current = 0)** field above.
- If not already specified (for write off), an **Adj. Type (Synch Current)** (on the Main page) is also required. It is used to synchronize (make equal) the current amount with the payoff amount when the contract is removed from (i.e. no longer covered by) a non-billed budget.

NBB Recommendation Rules

Set up any *NBB Recommendation Rules* that you want to be available for your non-billed budget contracts. Use any Non-billed Budget Daily Amount Calculation algorithms defined above. Also use the Non-billed Budget Monthly Payment Schedule (*NBPS-MON*) algorithm types and specify the default parameters.

Non-billed Budget Contract Types

You must set up a *Contract types* for your non-billed budget contracts. You may need multiple non-billed budget Contract types if you have different business rules for different types of non-billed budgets, for example, if you have both monitored and unmonitored non-billed budgets or if you support non-billed budgets with different renewal requirements. The following points provide guidelines for creating a non-billed budget Contract type.

Contract Type - Main (NBB)

Service Type should reference something like "Miscellaneous Service".

Distribution Code should be the one you set up to book credits for non-billed budgets.

Revenue Class should be set to `N/A`. (Revenue classes are not applicable because non-billed budgets do not apply a rate and revenue classes are only relevant for contract types that use a rate.)

The **Payment Segment Type** should reference the `Normal Payment`.

Do Not Overpay should be on. Any excess payments should go to the overpayment Contract, not the non-billed budget Contract.

Late Payment Charge is not applicable and should not be turned on because non-billed budgets are not billed.

Adj. Type (Synch Current) should reference the Synch Balance for Non-billed Budget adjustment type created above.

Contract Type - Detail (NBB)

- **Special Role** is `Non-billed Budget`.
- **Adjustment Type (Xfer)** must be populated to indicate the adjustment to be used for transferring accumulated credit from the non-billed budget contract to the covered contracts. This field is not used for unmonitored non-billed budgets.
- **Renewal** may be optional, not allowed, or required depending on your business processes.
- If Renewal is required, specify the **Days Before Expiration for Renewal**.
- **Non-billed Budget Monitoring** must indicate whether the non-billed budget is monitored by the account debt monitor.

Contract Type - Billing (NBB)

Non-billed budget contracts do not get billed, so the **Eligible for Billing** flag should be off.

Additionally, the **Characteristic Premise Required** should not be checked for non-billed budgets.

Contract Type - Rate (NBB)

Non-billed budget contracts do not use rates, so the **Rate Required** flag should be off.

Contract Type - Adjustment Profile (NBB)

Adjustment Type Profile should reference the *adjustment type profile* for non-billed budgets (set up above).

Contract Type - Credit and Collections (NBB)

The **Debt Class** should reference an appropriate value consistent with your credit and collections rules. Typically, monitored non-billed budgets should be in their own debt class. Unmonitored non-billed budgets should have a **Debt Class** that is not `Eligible for Collection`.

The **Write Off Debt Class** is not applicable because the non-billed budget contains no debt to be written off. Reference a debt class such as N/A. Refer to *Defining Credit & Collections Options* for more information.

Contract Type - Algorithms (NBB)

The Contract type *algorithms* defined above must be set up:

- The Non-billed Budget Credit Transfer algorithm created above should be specified for the Bill Completion system event (not used on unmonitored non-billed budgets).
- The Break Non-billed Budget Contract algorithm created above should be specified for the Break NBB Contract system event.
- The Non-billed Budget Process Scheduled Payment algorithm created above should be specified for the Process NBB Scheduled Payment system event (not used on unmonitored non-billed budgets).
- The Non-billed Budget Contract Renewal algorithm created above should be specified for the Contract Renewal system event.
- If you created the Automatic Contract Activation algorithm above, it should be specified for the Contract Creation system event.

- The Non-billed Budget Contract Activation algorithm created above should be specified for the Contract Activation system event.
- If you created the Automatic Contract Stop algorithm above, it should be specified for the Contract Stop Initiation system event.
- The Stop Non-billed Budget algorithm created above should be specified for the Contract Stop system event.

Contract Type - NBB Recommendation Rule (NBB)

Add the recommendation rules defined above that are valid for contracts of this type. Also, indicate which recommendation rule should be used as the default.

NBB Background Processes

Ensure that the following background processes are scheduled:

- Non-billed Budget Scheduled Payment Processing (*NBBPS*)
- Non-billed Budget Scheduled Payment Automatic Payment Create (*NBBAPAY*)
- Contract Renewal (*SARENEW*)
- Stop Expired Contracts (*SAEXPIRE*)

Chapter 12

Defining Quotation Options

Topics:

This section describes tables that must be set up before quotations can be created.

Fastpath: For more information, refer to *The Big Picture of Quotations*.

Setting Up Contract Types For Quotes

The topics in this section describe additional setup responsibilities required on contract types that can have proposal contracts.

Note: Assumption. We have assumed that you've already designed your contract types for the services that you sell. If you haven't done this, refer to *Defining Contract Types*.

Enabling The Automatic Generation Of Billing Scenarios

As described under *Proposal Contracts Contain Billing Scenarios And Template Consumption*, a proposal contract requires billing scenarios before a quote detail can be generated. The system will automatically create billing scenarios when a proposal contract is created if you plug-in the appropriate `Proposal contract Creation` algorithm on each such *contract type*. Refer to *PSAC-CBS* for an example algorithm (note, this algorithm can be used to create billing scenarios for financial services contracts).

Enabling The Generation Of Simulated Bill Segments

As described under *Creating Quotes And Quote Details*, in order for the system to generate simulated bill segments for a proposal contract, you must plug-in a `Proposal Contract Bill Segment Generation` algorithm on the proposal contract's *Contract type*. Refer to *CBSP-AR* for an example algorithm that creates a simulated bill segments for each billing scenario linked to the proposal contract by calling rate application.

Note: Interval pricing contracts. If you have proposal contracts that require the derivation of interval profiles from other interval profiles, you must also plug-in another `Proposal Contract Bill Segment Generation` on the interval *contract type*. Refer to *CBSP-IDDRV* for an example algorithm that performs derivation. When you plug-in this type of algorithm, don't forget to use a sequence number less than the one that generates the simulated bill segments (see above); otherwise, the derived interval profiles won't exist when rate application is called.

Enabling The Creation Of A Real Contract When A Quote Detail Is Accepted

As described under *Accepting / Declining Quote Details*, in order for the system to create a real contract type when a proposal contract type is accepted, you must plug-in a `Proposal contract type Acceptance` algorithm on the proposal contract type's *contract type*. Refer to *PSAA-PS* for an example algorithm that creates a real contract type by copying the proposal contract type.

Setting Up Quote Route Types

Quote route types control how quotes are *routed* to customers and prospects. To define a quote route type, open **Admin Menu, Quote Route Type**.

Fastpath: Refer to *Printing Quotes* for more information about how quotes are routed to customers and prospects.

Description of Page

Enter a unique **Quote Route Type** and **Description** for every quote route type.

Quote Routing Method controls the type of information that may be defined when the respective **Quote Route Type** is selected on *Account - Person Information*. The following options are available:

- `Postal`. Use this method if the routing is via the postal service.
- `Fax`. Use this method if the routing is via fax.
- `Email`. Use this method if the routing is via email.

Note: The values for **Quote Routing Method** are customizable using the *Lookup* table. This field name is QTE_RTG_METH_FLG.

- The next two fields control how quotes that are routed using this method are *printed* (both in batch and online).
  - Use **Batch Control** to define the process that creates the flat file that is passed to your quote printing software. Refer to *Technical Implementation of Printing Quotes In Batch* for more information about these processes.
  - Use **Extract Algorithm** to define the plug-in that constructs the "flat file records" that contain the information merged onto quotes routed using this method. Refer to *Printing Quotes* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_QTE_RTE_TYPE*.

Setting Up Terms and Conditions

Each T&C is identified with a terms and condition code. To define a terms and conditions code, open **Admin Menu, Terms and Conditions**.

Note: T&C print order. The value of the T&C code controls the order in which the T&C appears on the printed quote. This means you should assign these codes in some type of structured format (e.g., 01...) if you would like them to appear in a certain order.

Description of Page

Enter a unique **Terms and Condition** code and **Description**. Use **Text** to describe the exact terms.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_TC*.

Setting Up Decline Reasons

A proposal contract decline reason must be supplied when a proposal contract is declined. Open **Admin Menu, Proposal Contract Decline Reason** to define your reason codes.

Description of Page

Enter an easily recognizable **Decline Reason** and **Description** for each proposal contract decline reason.

Where Used

Decline reasons are used when a *quote detail is declined*.

Setting Up Customer Classes For Quotes

An optional plug-in spot exists on *customer class* where you can introduce additional logic to be executed when a quote is completed for an account that belongs to this customer class. The base package comes supplied with a sample algorithm that creates a workflow process when a quote is completed (refer to the algorithm *QTEC-WP* for more information).

Chapter 13

Defining Case Management Options

Topics:

Case management functionality is a highly configurable tool your organization can use to manage many situations, including (but certainly not limited to) the following:

- a high-bill complaint,
- a bankruptcy,
- a customer's request for literature,
- an application for new service,
- a customer's rejection of a quote,
- ... (the list is only limited by your time and imagination)

Obviously the steps involved in the resolution of the above cases are very different. The topics in this section describe how to configure the system to manage your cases as per your organization's desires.

Note: Separate module. Please note that the Case Management functionality is associated with a separate module. If the `Case Management` module is not applicable to your business you may turn it off. Refer to *Turn Off A Function Module* for more information.

Fastpath: Refer to *Case Management* for a description of how end-users use cases.

The Big Picture Of Cases

The topics in this section provide background information about how to configure the system to support your case management requirements.

Case Type Controls Everything

Whenever a user creates a case, they must specify the type of case (e.g., high-bill complaint, literature request, etc.). The case type controls how the case is handled.

Case types hold the business rules that control cases. Since these business rules can sometimes be quite complicated, setting up case types requires planning and foresight. The topics in this section describe the type of business rules that can be configured on your case types.

Person or Account Applicability

Some types of cases may be person-oriented, others may be account-oriented. For example:

- Cases used to keep track of a literature request would reference the person who requested the literature.
- Cases used to keep track of a high-bill complaint would reference the account associated with this bill(s) being disputed.

When you set up a case type, you define if its cases must reference a person and/or account. Note, any combination of these objects is permitted on a case.

Contact Information Applicability

When a case is created, you may want to keep track of how to contact its originator. For example, you may want to record the originator's email address or phone number. When you set up a case type, you define if contact information is required, optional or not allowed on its cases.

Business Object Association

A case type may reference a Business Object, which serves as a link between cases of that type and the options that are associated with the business object.

Additional Information

Some of your cases may require additional information (in the form of *characteristics*). For example, a high-bill complaint may require at least one bill. When you set up a case type, you can define the additional fields that are required. In addition, you can define default values for these fields.

The case functionality also allows you to require characteristics when a case enters a given state. Refer to *Required Fields Before A Case Enters A State* for the details.

Note: Requiring supporting documents. Because any *type of characteristic* can be referenced on a case, you can require references to supporting documents by requiring a `file location` characteristic.

Access Rights

You can take advantage of the system's *security* to restrict cases of a given type to certain users. For example, you can restrict high-bill complaints to specific user groups.

The following points describe how to implement this type of security:

- Create an *application service* for each type of case you need to secure.
- Define the access modes `Add`, `Inquire` and `Change` for each application service.
- Define the applicable application service on each case type.
- Link the appropriate *user groups* to each application service.

- For user groups that are allowed to add cases of a given type, define `Add` as a valid access mode.
- For user groups that are allowed to view cases of a given type, define `Inquire` as a valid access mode
- For user groups that are allowed to change cases of a given type, define `Change` as a valid access mode

If you restrict access to a case type's cases, you can further restrict which users can work on cases given the status of the case. Refer to *Which Users Can Transition A Case* for more information.

Note: Restricting access to cases is optional. If you don't specify an application service on a case type, all users (who have access to the case transaction) may access its cases.

Lifecycle

Many objects in the system have predefined lifecycles whose rules are governed by the base-package and cannot be changed. For example, a contract starts out in the `Pending Start` state and eventually becomes `Closed` when it's been final billed (and paid). You can't change the system to allow a contract to starts its life in the `Closed` state.

The lifecycle of cases is *not* governed by the base package. Rather, you define the lifecycle of your cases when you set up their case types. Examine the following diagrams; the one on the left shows the potential lifecycle of a case that manages a high-bill complaint, the one on the right shows the potential lifecycle of a case that manages a customer's literature request.

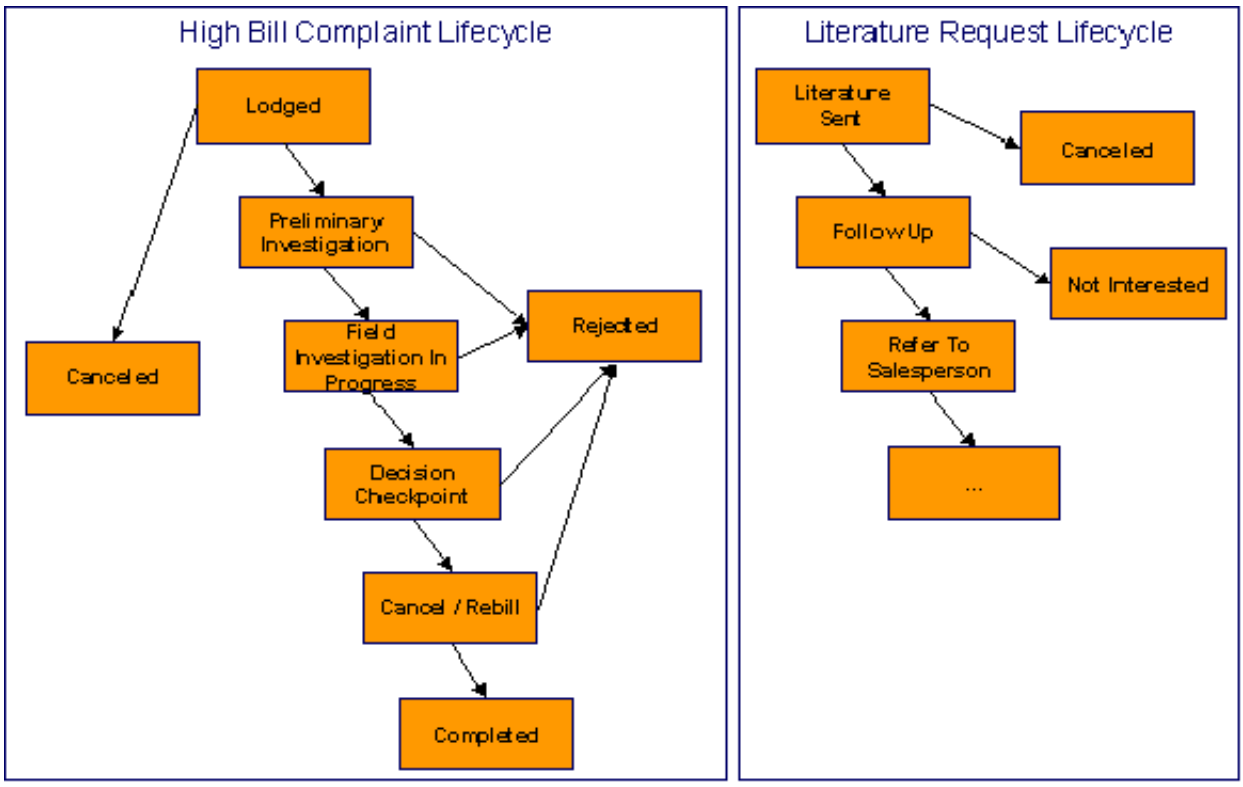


Figure 5: Potential Lifecycles Of Two Types Of Cases

Note: Just examples. The above lifecycles are just examples. When you set up your case types, you must define the valid states for your case type.

The topics that follow describe important concepts that are illustrated in the above diagrams.

Valid States versus State Transition Rules

The orange boxes in the above diagram show the potential valid states a given case can have. The lines between the boxes indicate the state transition rules. These rules govern the states a case can move to while it's in a given state. For example, the above diagram indicates a high bill complaint that's in the `Lodged` state can be either `Canceled` or moved into the `Preliminary Investigation` state.

When you set up a case type, you define both its valid states and the state transition rules.

Transitory States

You can define a state in a case type as **Transitory** if you do not wish the case to exist in a particular state.

One Initial State and Multiple Final States

When you set up a case type's states, you must pick one as the initial state. The initial state is the state assigned to new cases of a given type. For example, high-bill complaint cases have an initial state of `Lodged`, whereas literature request cases have an initial state of `Literature Sent`.

You must also define which statuses are considered to be "final". When a case enters a "final" state, it is complete and no further action is necessary. You might want to think of the "final" states as the potential outcomes of a case. For example, a high-bill complaint has potential outcomes of `Completed`, `Rejected`, and `Canceled`.

The "final" states are used by the system to differentiate between open and closed cases. For example, an alert highlights when the person / account in context has open cases (this alert only exists if you've plugged-in the appropriate installation *alert*).

Allowing A Case To Be Reopened

You can set up your state transition rules to allow a case to be reopened (i.e., to be moved from a final state to a non-final state). Neither of the above examples allows this, but it is possible if you configure a case type accordingly.

Make Sure To Have A Canceled State

The system does not allow you to delete a case. Therefore, if you want to support logical deletion, you should have a status of `Canceled` early in a case type's lifecycle. Doing this allows a user to cancel (i.e., logically delete) a case.

Note: Cancel reason. You might want to consider setting up your case types to require a cancel reason (in the form of a *predefined value characteristic*) when a user cancels a case. Refer to *Required Fields Before A Case Enters A State* for more information.

Buttons Are Used To Transition A Case From Status To Status

When a case is displayed on *Case - Main*, a separate button is shown for each state into which the case can be transitioned. For example, a high-bill complaint case that is in the `Lodged` state would show two buttons: **Start Investigation** and **Cancel**. If the user presses the **Start Investigation** button, the case is transitioned to the `Preliminary Investigation` state. If the user presses the **Cancel** button, the case is moved to the `Canceled` state.

You may define the text displayed on the button differently for each state transition. This allows the action description to be varied according to the previous status. For example, the button to transition from `New` to `Active` may be labeled **Activate**, but the button to change from `Closed` to `Active` may be labeled **Reactivate**.

Refer to *Which Users Can Transition A Case* for instructions describing how to restrict users to specific actions.

State Transitions Are Audited

The system maintains an audit trail whenever a case transitions from one state to another. This audit is shown in the case's *log*.

Reports and Analytics Highlight Productivity

When you set up a case type's lifecycle, keep in mind that several reports and analytics highlight how long it took cases to transition into a state. For example, you can use a report to see how long it took high-bill complaints to be completed (or initially actioned or...). Refer to the *Reports* chapter for the details of case reports.

Status-Specific Business Rules

As described in *Lifecycle*, when you set up a case type, you define the possible states its cases can pass through. The following topics describe business rules that can be configured for *each state*.

A Script That Helps A User Work Through A Case

You can define a *Business Process Assistant script* that helps a user work a case while it's in a given state. For example, when you set up the `Preliminary Investigation` state for the high-bill complaint case type, you can define a script. A user can then easily launch this script to help them work through a case in this state.

Please keep the following in mind when you're designing how to integrate BPA scripts with your cases:

- You can have a different script for each state. For example, you could develop a script to help a user work on a case while it's in the `Preliminary Investigation` state and a different script to help them work in a case while it's in the `Decision Checkpoint` state.
- Rather than make a user launch a script by pressing a hyperlink on the *case page*, you can have the system automatically launch the script while the case is in a given state. Refer to *Script Launching Option* for more information.
- You can also have the system automatically launch a script when a user selects a ToDo entry. Refer to *Launching Scripts When ToDo Entries Are Selected* for more information.

Fastpath: Refer to *Scripts and Cases* for more information about how to streamline your case processing with scripts.

Required Fields Before A Case Enters A State

You can define additional fields (i.e., characteristics) that are required before a case can enter a given state. For example,

- You can indicate a high-bill complaint must reference at least one bill before it enters the `Preliminary Investigation` state.
- You can indicate a case must reference a cancel reason before it enters the `Canceled` state.

You do this by indicating that *characteristics* (that were optional when the case was added) are required when a case enters a given state.

Validation Before A Case Enters A State

You can define validation that executes before a case can enter a given state. For example, you can indicate the case must have been assigned a responsible user before it can enter the `Preliminary Investigation` state. This validation logic is held in algorithms that are plugged in on the case type and therefore you can define any type of validation.

Additional Processing When Entering A State

You can define additional processing that should happen when a case enters a given state. For example, you can have a *letter* created when a high-bill complaint case is `Rejected`. Similarly, you can have a *To Do entry* created when a high bill complaint enters the `Preliminary Investigation` state. This additional processing is held in algorithms that are plugged in on the case type and therefore you can define any type of additional processing.

You can also incorporate state transitioning logic within routines that are executed when a case enters a state, so that you do not need to rely upon CASETRAN to transition your cases.

Validation Before A Case Exits A State

You can define validation that executes before a case can exit a given state. For example, you might want to check the account's balance is less than a given value before a case can exit a given state. This validation logic is held in algorithms that are plugged in on the case type and therefore you can define any type of validation.

Additional Processing When Exiting A State

You can define additional processing that should happen when a case exits a given state. For example, you can have a *To Do entry* automatically completed when a high bill complaint leaves the `Decision Checkpoint` state. This additional processing is held in algorithms that are plugged in on the case type and therefore you can define any type of additional processing.

Automatic Transition Rules

You can define rules that automatically transition a case into a different state. For example, you can indicate a literature request should be transitioned to the `Follow Up` state 1 week after the literature is sent. These rules are held in algorithms that are plugged in on the case type and therefore you can define any type of automatic transition rules.

Cases in a state with automatic transition rules are monitored by the *CASETRAN* background process. Each time this program runs, the respective automatic transition plug-in is called for each such case and it transitions the case if the condition applies.

When the user adds a new case or changes the state of a case manually the system attempts to auto-transition the case to subsequent statuses as necessary. If auto-transition rules apply to the new state (and to subsequent ones) they would be executed right away. In other words, you don't need to wait for the auto-transition background process to be executed. An indication that the case was auto-transitioned online is displayed right below the action buttons section.

Note: Auto-Transition Errors. Online auto-transition is performed recursively committing each successful state transition to the database. It is performed up to 100 times or until an error is encountered during the process. If this happens, auto-transition stops at the last **non-transitory** state into which a successful transition had occurred. Two case log entries will be generated automatically - one containing the message that a transition error has occurred, and a second containing the actual error message. A To Do entry will also be generated automatically upon rollback. The type of this To Do entry will be taken from 1) the `Case Transition Exception To Do Type` **option** for the **Business Object** associated with the case type, and if this is not populated, 2) the `Exception To Do Type` indicated on the Case Options Feature Configuration. All of the above error handling is true for both batch and online processing of cases.

Note: Triggering Auto-Transition. If you have a customized process that affects the state of a case and you want the case to be auto-transitioned right away, i.e. not wait for the next scheduled *CASETRAN* background process to execute, you can customize that process to trigger auto-transition for the specific case, or you can put the state transition logic into the routines that execute at state entry time.

Script Launching Option

You can define whether the script associated with a given state is to be automatically launched while the case is in that state. The system supports the following options:

- Launch the script only if no script is currently active.
- Always launch the script unless this specific script is currently active.

Caution: With this option, if a script is currently open in the page's BPA script area then it will be automatically closed and the case script will open.

- Do not automatically launch the script.

You do this by plugging-in a `Script Launching` algorithm for the given state. If no such plug-in is provided the script is not automatically launched.

Which Users Can Transition A Case Into A State

If you have *restricted access* to a case type, you can further restrict which user groups are allowed to transition a case into specific states. For example, you can control which user group can transition a high bill complaint into the `Preliminary Investigation` state. The following points describe how this is done:

- Define actions on the *application service* defined on the case type. You must define an action for each status that you need to secure.
- Define each status's corresponding action. Note, you only need to link a status to an action if it's secured. Any user with *access* to the case type can perform statuses that aren't linked to actions.
- Define the transition role for each status's valid next status. You can assign valid next statuses to be reachable via system (only), or system and user.

- Define which *user groups* have access to the actions (i.e., statuses). In addition, these user groups should have access to the `Change` action.

Responsible User Applicability

Some of your cases may require a "responsible user". This is the user who has overall responsibility for the case. When you set up a case type, you define if a responsible user is required, optional or not allowed on its cases.

The following points describe how to set up the system if a responsible user is not required when a case is first created, but is later in its lifecycle:

- Indicate that a responsible user is optional on the case type
- Plug-in either an *exit validation* or *entry validation* algorithm on one of the case type's states to require a responsible user at some point in a case's lifecycle

Note: Address To Do entries to the responsible user. If you use the *base-package algorithm* to create a To Do entry when a case enters a given state, you can indicate that the To Do entry should be addressed to the responsible user on the case.

Scripts and Cases

There are three ways *Business Process Assistant scripts* can be used to manage cases:

- You can create a BPA script to help users create a case. For example, a script can help a user create a new high-bill complaint.
- Using a script to create a case can save a user a lot of time (and training efforts). This is because the script can automatically populate many fields on the case based on answers to questions.

Refer to *Initiating Scripts* for a description of how end-users initiate scripts.

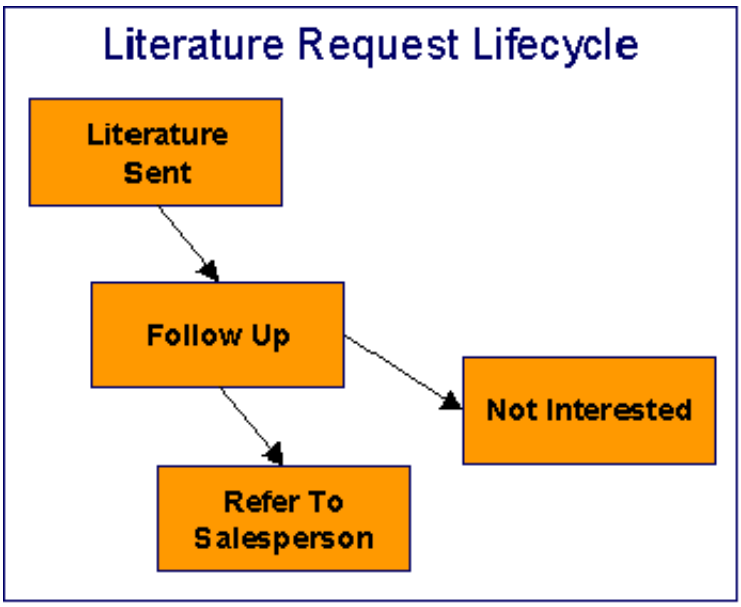
- You can create a script to help users work on a case when it's in a given state. Refer to *A Script That Helps A User Work A Case* for more information.
- You can *set up your case types to create ToDo entries* to notify users when cases exist that require their attention. Users can complete many of these ToDo entries without assistance. However, you can set up the system to automatically launch a script when a user selects a ToDo entry. For example, consider a ToDo entry that highlights a high-bill complaint that requires investigation. You can set up the system to execute a specific script when a user selects this ToDo entry. This script might guide the user through the investigation process (and help them update the case). Refer to *Executing A Script When A To Do Entry Is Selected* for more information.

To Do's and Cases

The topics in this section provide background information about how to facilitate case management with *To Do entries*.

Creating and Completing To Do Entries

You can configure your case types to create and complete *To Do entries* when a case enters or exits a state. Let's use the following *lifecycle diagram* to illustrate a potential use of ToDo's:



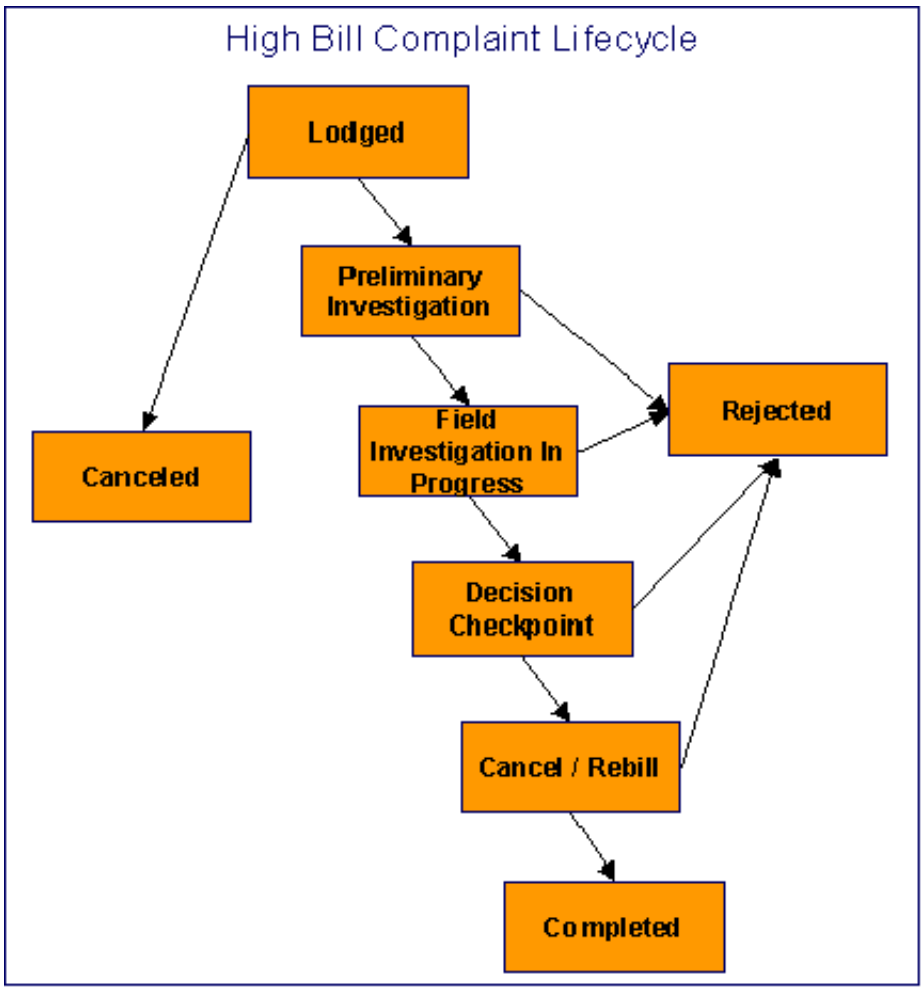
Let's assume the following:

- You want a ToDo entry created when a literature request case enters the `Follow Up` state. You want this ToDo automatically completed when the case enters either the `Refer To Salesperson` or `Not Interested` states. Note, we refer to this as the "first" ToDo entry below.
- You want a different ToDo entry created when a case enters the `Refer To Salesperson` state. You do not want the system to automatically complete this entry (the sales person must manually do this). Note, we refer to this as the "second" ToDo entry below.

To implement the above, you would set up a case type as follows:

- Plug-in an *entry processing* algorithm on the `Follow Up` status to create the first ToDo entry.
- Plug-in an *exit processing* algorithm on the `Follow Up` status to complete the first ToDo entry.
- Plug-in an *entry processing* algorithm on the `Refer To Salesperson` status to create the second ToDo entry.

While the case type illustrated above had a single ToDo entry "active" at any point in time, you can easily configure a case type to have multiple ToDo entries active at any point in time. Let's use the following lifecycle diagram to illustrate this point:



Let's assume the following:

- You want a ToDo entry created when a high bill complaint is created and you want it completed when the case reaches the `Canceled`, `Rejected` or `Approved` states. This ToDo entry could be used by a supervisor to monitor the number of high-bill complaints being worked. Note, we refer to this as the "first" ToDo entry below.
- You want a different ToDo entry created when the case enters the `Preliminary Investigation` state and you want this entry automatically completed when the case leaves this state. Note, we refer to this as the "second" ToDo entry below.
- You want a different ToDo entry created when the case enters the `Decision Checkpoint` state and you want this entry automatically completed when the case leaves this state. Note, we refer to this as the "third" ToDo entry below.

To implement the above, you would set up the case type as follows:

- Plug-in an *entry processing* algorithm on the `Lodged` status to create the first ToDo entry. Plug-in an *entry processing* algorithm on the `Canceled`, `Rejected` and `Completed` statuses to complete this entry.
- Plug-in an *entry processing* algorithm on the `Preliminary Investigation` status to create the second ToDo entry. Plug-in an *exit processing* algorithm onthe `Preliminary Investigation` status to complete this entry. We elected to use an exit processing algorithm because we only have to plug it in on one status. If we'd used an entry processing algorithm, we would need to plug it in on the 2 statuses into which a `Preliminary Investigation` status can transition.

- Plug-in an *entry processing* algorithm on the `Decision Checkpoint` status to create the third ToDo entry. Plug-in an *exit processing* algorithm onthe `Decision Checkpoint` status to complete this entry.

Launching Scripts When To Do Entries Are Selected

You can set up your case types to create ToDo entries to notify users when cases exist that require their attention. Users can complete many of these ToDo entries without assistance. However, you can set up the system to automatically launch a script when a user selects a ToDo entry. For example, consider a ToDo entry that highlights a high-bill complaint that requires investigation. You can set up the system to execute a specific script when a user selects this type of ToDo entry. This script might guide the user through the investigation process. Refer to *Executing A Script When A ToDo Entry Is Selected* for more information.

All To Do Entries Are Visible

When a case is displayed on *Case Maintenance*, the system summarizes the number of ToDo entries associated with the case (if you've *set up your ToDo types* appropriately).

Setting Up Case Management Options

The topics in this section describe how to set up the system to enable case management.

Caution: The following topics assume you thoroughly understand the concepts described under *The Big Picture Of Cases*.

Installation Options

Case Info May Be Formatted By An Algorithm

An algorithm may be plugged in on the *installation record* to format the standard case information that appears throughout the system. Refer to *CSIN-DFLT* for an example of this algorithm.

This algorithm may be further overridden by a "Case information" plug-in on the *Case Type*. Refer to *C1-CT-INFO* for an example of this algorithm.

Alert Info Is Controlled By An Installation Algorithm

An algorithm that is plugged in on the *installation record* is responsible for formatting the alerts that highlight if the person / account in context has open cases. Refer to *CCAL-CASE* for an example of this algorithm.

Setting Up Application Services

As described under *Access Rights*, you can prevent unauthorized users from accessing cases. The following points describe how to implement this type of security:

- Create an *application service* for each case type that needs to be secured
- Create an action on the application service for each status you need to secure
- Link the valid *user groups* to the application service and define which actions they can perform
- Define the application service on the *case type*
- Define the related action for each status on the *case type / status*

Setting Up Scripts

As described under *Scripts and Cases*, BPA scripts can facilitate the creation and working of cases. Refer to the *Defining Script Options* for instructions describing how to set up scripts.

Setting Up To Do Types

As described under *ToDo's and Cases*, ToDo entries can be used to highlight cases that require user attention.

The following points provide a high-level description of how to create (and complete) ToDo entries for a case type:

- Create a ToDo type for each different type of ToDo entry used during a case's lifecycle.
  - On the ToDo type, think carefully about the roles whose users can work on the entries
  - Also consider if you would like a BPA script launched when a user selects the entry
- Specify the ToDo type on the appropriate *entry processing* or *exit processing* algorithm.
- If you want the system to automatically complete ToDo entries, specify the ToDo type on the appropriate *entry processing* or *exit processing* algorithm.

Please be aware that the case maintenance transaction highlights the number of open and being worked ToDo entries linked to the case being displayed on the page. However, the system can only do this if the ToDo entries reference a *foreign-key characteristic* whose foreign key references the case table. If you use the *CSEN-TD* algorithm to create ToDo entries when a case enters a given state, this algorithm will do this for you if:

- You have set up a *foreign-key characteristic type* whose *foreign key* references the case table.
- In addition, the characteristic type must reference a characteristic entity of `ToDo Entry`.

Setting Up Characteristic Types

As described under *Additional Information*, some of your cases may require additional information (in the form of *characteristics*). If this is true, you must set up the characteristic types before setting up the case types.

Refer to *Setting Up ToDo Types* for instructions regarding a characteristic type that must be set up in order for the system to know the ToDo entries that are associated with a case.

If you use the *CSEN-CC* algorithm to create customer contacts when a case enters a given state, you should set up a *foreign-key characteristic type* as follows:

- Its *foreign key* must reference the case table
- In addition, the characteristic type must reference a characteristic entity of `Customer Contact`

Setting Up Case Types

The case type maintenance transaction is used to maintain your case types. The topics in this section describe how to use this transaction.

Fastpath: Refer to *The Big Picture Of Cases* for more information about how a case type encapsulates the business rules that govern a case.

Case Type - Main

Use this page to define basic information about a case type.

Open the case type page by selecting **Case Type** from the admin menu.

Main Information

Enter a unique **Case Type** code and **Description** for the case type.

Use **Long Description** to provide a more detailed explanation of the purpose of the case type.

Person Usage controls the applicability of a person on cases of this type. Select `Required` if a person must be defined on this type of case. Select `Optional` if a person can optionally be defined on this type of case. Select `Not Allowed` if a person is not allowed on this type of case.

Account Usage controls the applicability of an account on cases of this type. Select `Required` if an account must be defined on this type of case. Select `Optional` if an account can optionally be defined on this type of case. Select `Not Allowed` if an account is not allowed on this type of case.

If you need to restrict access to cases of this type to specific user groups, reference the appropriate **Application Service**. Refer to *Setting Up Application Services* for the details of how to secure access to your cases.

If you are configuring a case type to handle the processing of data defined via a **Business Object**, associating the case type with a business object serves to link the properties of the business object (e.g. BO options) with cases of that type. Refer to *Business Objects* for further information. In addition, refer to *Automatic Transition Rules* for information on the role of BO options in case auto-transition errors.

Responsible User Usage controls the applicability of a responsible user on cases of this type. Select `Required` if a responsible user must be defined on this type of case. Select `Optional` if a responsible user can optionally be defined on this type of case. Select `Not Allowed` if a responsible user is not allowed on this type of case. Refer to *Responsible User Applicability* for more information.

Contact Information Fields

There are three contact information fields: **Contact Person & Method Usage**, **Contact Instructions Usage**, and **Callback Phone Usage**. These fields are used to determine whether or not each type of contact information must be entered on case records with this case type. Select `Required` if the contact information must be entered, select `Optional` if the user can choose whether or not to include the contact information on this type of case, or select `Not Allowed` if the contact information cannot be entered on this type of case.

Algorithms

The **Algorithms** grid contains algorithms that control functions for cases of this type. You must define the following for each algorithm:

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

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

System Event	Optional / Required	Description
`Case Information`	Optional	We use the term "Case information" to describe the basic information that appears throughout the system to describe a case. The data that appears in "case information" is constructed using this algorithm. Plug an algorithm into this spot to override the "Case information" algorithm on installation options or the system default "Case information" if no such algorithm is defined on installation options. Click *here* to see the algorithm types available for this system event.

Case Type Tree

The tree summarizes the case type's lifecycle. You can use the hyperlinks to transfer you to the **Lifecycle** tab with the corresponding status displayed.

Case Type - Case Characteristics

To define characteristics that can be defined for cases of this type, open **Admin Menu, Case Type** and navigate to the **Case Characteristics** tab.

Description of Page

Use **Sequence** to control the order in which characteristics are defaulted. Turn on the **Required** switch if the **Characteristic Type** must be defined on cases of this type. Turn on the **Default** switch to default the **Characteristic Type** when cases of this type are created. Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** box is checked. Refer to *Required Fields Before A Case Enters A State* for a description of how you can make option characteristics required at later stages in a case's lifecycle.

Case Type - Lifecycle

Case types that involve multiple users and multiple potential outcomes have complex lifecycles. Before you can design a case type's lifecycle, it's important that you thoroughly understand the concepts described under *Lifecycle* and *Status-Specific Business Rules*. After thoroughly understanding these concepts, we recommend you perform the following design steps:

- Draw a "state transition diagram" as illustrated above under *Lifecycle*. Keep in mind that if your state transition diagram is complex, your cases will be complex. While some cases warrant complexity, you should always ask yourself if there aren't better ways to achieve the desired results if your first effort results in complexity.
- Determine which characteristics (if any) are required during each stage of a case's lifecycle
- Determine when ToDo entries (if any) should be created (and completed) during a case's lifecycle
- Determine additional validation (if any) that should be executed before a case enters and exits each state
- Determine additional processing (if any) that should transpire when a case enters or exits each state
- Determine if scripts are warranted to help users work the cases and, if so, design the scripts for each applicable state

When the above tasks are complete, you will be ready to set up a case type's lifecycle.

Open the Lifecycle page by selecting **Case Type** from the admin menu then the **Lifecycle** tab.

Note: You can navigate to a status by clicking on the respective node in the tree on the Main tab. You can also use the hyperlinks in the Next Statuses grid to display a specific status in the accordion.

Main Information

The **Status** accordion contains an entry for every status in the case type's *lifecycle*.

Use **Status** to define the unique identifier of the status. This is NOT the status's description, it is simply the unique identifier used by the system.

Use **Description** to define the label that appears on the lifecycle accordion as well as the status displayed on the case.

Use **Script** to reference a BPA script that can assist a user work on a case while it's in this status. Refer to *A Script That Helps A User Work Through A Case* for the details.

Use **Access Mode** to define the action associated with this status. This field is disabled if an application service is not specified on the Main page. Refer to *Access Rights* for the details of how to use this field to restrict which users can transition a case into this state.

Use **Batch** to specify a batch control that will auto-transition the case. Any case in a status configured with a batch control will be transitioned when the batch job runs (rather than when *CASETRAN* is executed). For this purpose, batch process *C1-CSTRS* (**Case Scheduled Transition**) is supplied with base package, which will execute all Exit Status logic for the current status, and Enter Status logic for the destination status. You may choose to create a batch process with your own transition logic.

Note: If you wish to defer transitioning a case in a particular status until the batch process on your case type status is executed, you should not populate an Auto-Transition algorithm on that status. Otherwise, CASETRAN will transition the case according to your Auto-Transition logic.

Use **Comment** to describe the status. This is for your internal documentation requirements.

Use **Sequence** to define the relative order of this status in the tree on the Main page.

Use **Status Condition** to define if this status is an `Initial`, `Interim` or `Final` state. Refer to *One Initial State and Multiple Final States* for more information about how this field is used.

Use **Transitory State** to indicate whether a case should ever exist in this state. Only `Initial` or `Interim` states can have a transitory state value of `No`.

The **Alert Flag** is used to indicate whether or not an alert should be displayed for customers with cases in the state. (The alert is shown via the base package Installation - Alert algorithm.)

Algorithms

The **Algorithms** grid contains algorithms that control important functions for cases of this type. You must define the following for each algorithm:

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

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Description
`Auto Transition`	This algorithm is executed to determine if a case that's in this state should be transitioned into another state. Refer to *Automatic Transition Rules* for the details. Click *here* to see the algorithm types available for this system event.
`Enter Processing`	This algorithm holds additional processing that is executed when a case is transitioned into this state. You can also specify state transition logic within Enter Processing routines. Refer to *Additional Processing When Entering A State* for the details. Click *here* to see the algorithm types available for this system event.
`Enter Validation`	This algorithm holds validation logic that executes before a case can enter a given state. Refer to *Validation Before A Case Enters A State* for the details. Click *here* to see the algorithm types available for this system event.

System Event	Description
`Exit Processing`	This algorithm holds additional processing that is executed when a case is transitioned out of this state. Refer to *Additional Processing When Exiting A State* for the details. Click *here* to see the algorithm types available for this system event.
`Exit Validation`	This algorithm holds validation logic that executes before a case can by transitioned out of a given state. Refer to *Validation Before A Case Exits A State* for the details. Click *here* to see the algorithm types available for this system event.
`Script Launching`	This algorithm sets the script launching option for the script associated with a given state, if any. Refer to *Script Launching Option* for the details. Click *here* to see the algorithm types available for this system event.

Next Statuses

Use the **Next Statuses** grid to define the statuses a user can transition a case into while it's in this state. Refer to *Valid States versus State Transition Rules* for more information. Please note the following about this grid:

- Use **Action Label** to indicate the verbiage to display on the action button used to transition to this status.
- **Sequence** controls the order of the buttons that appear on *Case - Main*. Refer to *Buttons Are Used To Transition A Case Into A State* for more information.
- **Use as Default** controls which button (if any) is the default button.
- **Transition Condition** may be configured to identify a common transition path for cases of this type in the current state. This transition condition may then be referenced across multiple case types. You'll need to add values to Look Up table field **TR_COND_FLG** that fit the typical transitions for your case types (e.g. `Ok`, `Error`, etc.).

By assigning the transition condition value to a given "next status", you can design your Enter State transition or Auto-Transition logic to utilize those flag values *without specifying a status particular to a given case type*. Thus, similar logic may be used across a range of case types to transition a case into, for example, the next `Ok` state for the case's current status.

- **Transition Role** controls whether only the `System` or both `System and User` have the ability to transition a case into a given "next status".
- You can use the status description hyperlink to open the Status accordion to the respective status.
- When you initially set up a case type, none of the statuses will reside on the database and therefore you can't use the search to define a "next status". We recommend working as follows to facilitate the definition of this information:
  - Leave the Next Statuses grid blank when you initially define a case type's statuses
  - After all statuses have been saved on the database, update each status to define its Next Statuses (this way, you can use the search to select the status).

Required Characteristics

Use the **Required Characteristics** grid to define characteristics that are required when a case enters this state. Only `Optional` characteristics defined on the main page appear in this grid. Refer to *Required Fields Before A Case Enters A State* for more information.

Chapter 14

Workflow and Notification Options

Topics:

We use the term "notification" to reference the electronic transactions that you exchange with third parties when:

- They need information about a customer
- They need to change something about a customer

For example, a financial service provider sends a notification to a financial services company when a customer elects to use them as their financial service provider.

When a notification is received, the system responds by creating a workflow process. The workflow process contains workflow events. These events perform the processing necessary to execute the notification.

Workflow processing is difficult to explain because its flexible design can be used to automate many different types of multi-event processes. For example, you can use workflow processing to manage the events.

Caution: Setting up the workflow process control tables is as challenging as your organization's business rules. If you don't have automated workflow processes, you don't have to setup anything. If you have sophisticated workflow processing requirements, your setup process will be more challenging.

The topics in this section describe tables that control your automated workflow and notification processing.

The Big Picture Of Workflow Processing

Fastpath: Refer to *The Lifecycle Of A Workflow Process And Its Events* for more information about workflow processes.

The Big Picture Of Workflow Events

This section describes the various types of workflow events and their lifecycle:

How Are Workflow Events Created?

Workflow events may be created as follows:

- The process that uploads notification requests creates a workflow process to implement the notification request. The workflow process has one or more workflow event(s). The number and type of events is controlled by the workflow process template associated with the workflow process. Refer to *What Type Of Workflow Process Is Created?* for more information about how notification requests cause workflow processes to be created.
- Workflow events will be created when an operator creates an ad hoc workflow process. The number and type of workflow events are controlled by the workflow process template associated with the workflow process.
- An ad hoc workflow event may be created and linked to an existing workflow process by an operator at their discretion.
- The system may be customized to create a workflow process when something noteworthy happens in the system. Refer to *System Conditions May Trigger Notification and Workflow* for more information.

Note: Bottom line. Most workflow events are created by the system when it creates a workflow process to implement an incoming notification. If you need to create an ad hoc workflow event, you can either create a workflow process using a template that contains the desired event OR link the desired event to an existing workflow process.

Fastpath: For more information about creating ad hoc workflow processes and events, refer to *Workflow Process Maintenance*.

Executing Workflow Events On Their Trigger Date

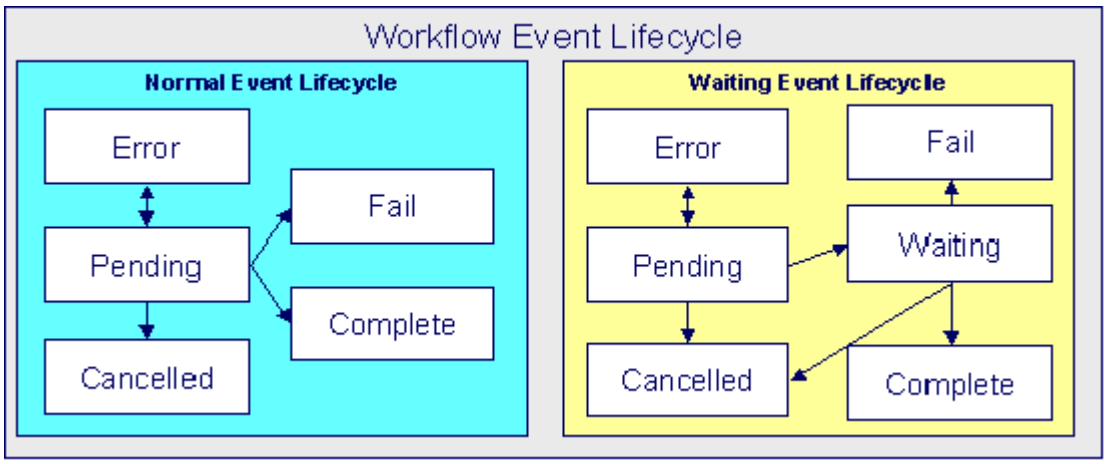
When the background process (referred to as `WFET`) executes a workflow event on its trigger date, it calls the activation algorithm associated with the event's event type. Because you can add and change activation algorithms at will, the variety of workflow events is infinite.

Fastpath: Refer to *Designing Workflow Event Types* for more information about activation algorithms.

Workflow Event Lifecycle

The lifecycle of a workflow event is dependent on whether the event has to wait on something before it can complete or fail.

The following diagram shows the possible lifecycle of a workflow event:



The following points explain the lifecycle of workflow events of the *normal* variety:

- *Normal* events are initially created in the `pending` state.
- On a `pending` event's trigger date, the workflow event activation process (referred to as `WFET`) executes the event's activation algorithm.
- An event's activation algorithm may cause a `pending` event to become in `error`. A user should correct the cause of the error and then change the event's status back to `pending`. The `pending` event will be processed the next time the activation program runs.
- An event's activation algorithm may cause a `pending` event to `fail`. For example, if an event used to validate a notification detects invalid information (e.g., an incoming notification is missing the customer's account number), the event will `fail`.
- A `pending` event becomes `complete` when the event's activity is successful. For example, if an event used to validate a notification determines the notification is valid, the event will `complete`. A user may manually change the status to `complete` if the event type indicates that manual completion is allowed.

Fastpath: For more information about a workflow event's trigger date, refer to *Workflow Event Dependencies & Trigger Date*.

- A `pending` event will be `cancelled` automatically by the system if the workflow *process* is cancelled (a workflow process will be `cancelled` if an event fails or at the discretion of an operator). A user may `cancel` an event at will. Refer to *How Are Workflow Events Canceled* for more information.

The following points explain the lifecycle of workflow events of the *waiting* variety:

- *Waiting* events are initially created in the `pending` state.
- On a `pending` event's trigger date, the system executes the event.
- An event's activation algorithm may cause a `pending` event to become in `error`. A user should correct the cause of the error and then change the event's status back to `pending`. The `pending` event will be processed the next time the activation program runs.
- If the activation algorithm did not cause the event to become in `error`, the event's status is changed to `waiting`.

Fastpath: For more information about a workflow event's trigger date, refer to *Workflow Event Dependencies & Trigger Date*. For more information about what an event might wait on, refer to *Waiting Events And Their Waiting Process*.

- A `waiting` event becomes `complete` when the system sees that the thing that it's waiting for is finished.
- A `waiting` event `fails` when the system sees that the thing that it's waiting for didn't complete successfully.
- A `waiting` event will be `cancelled` automatically by the system if the workflow process is cancelled (a workflow process will be `cancelled` if an event fails or at the discretion of an operator).
- A `pending` event will be `cancelled` automatically by the system if the workflow process is cancelled (a workflow process will be `cancelled` if an event fails or at the discretion of an operator).

Fastpath: Refer to *Workflow Processes Can Have Multiple Branches* for more information about event transition in a process with multiple branches.

Workflow Event Dependencies & Trigger Date

The trigger date of most workflow events is blank when they are first created. This is because the trigger date can only be set when ALL of the preceding workflow events *on which it depends* are complete. An example will help explain why this design is necessary. Consider the following example that shows a simple workflow process and its events:

Event Number	Workflow Event	Dependent On Event(s)	Trigger Date Set To X Calendar Days After Completion Of Preceding Events
10	Validate new customer notification	N/A - first event	0
20	Set up new customer	10	0
30	Send notification confirming new customer	20	0
40	Send welcome letter	20	0

This workflow process is meant to implement the following:

- On the first day, validate the incoming notification (the one that tells us about a new customer).
- If validation is successful, set up the new customer in the system.
- After the new customer is set up, send a notification to the requester that everything has been set up. Also, send a letter to the customer.

The problem is that you don't want to execute event 20 until event 10 is complete. This is achieved by indicating event 20 is dependent on event 10. The system will only populate event 20's trigger date when event 10 is complete. Similarly, you can't set the trigger dates of events 30 and 40 unit the customer has been set up (event 20). So, when you set up a workflow event, you must indicate the dependent events. If you only want the next event to trigger X days after the completion of earlier events, you can indicate such.

Note: Bottom line. The trigger date of a workflow event is set to the current date plus X days where X is the number of calendar days defined on the workflow event. If this date is not a workday for your organization, the trigger date will be set to the next workday. If the resultant date is the current date (because X is zero), the event will be activated immediately.

Fastpath: Refer to *How Are Workflow Events Completed?* for information about how these events are executed.

Workflow Events May Be In Error

As explained under *Workflow Event Lifecycle*, when the background process `WFET` executes an event's activation algorithm, this algorithm may cause a `pending` event to become in `error`.

For every workflow event that's in error, a record is written to the *workflow event exception* table. To view the errors,

A user should correct the cause of the error and then change the event's status back to `pending`. The `pending` event will be processed the next time the activation process runs.

Fastpath: Refer to *Errors versus Failure* for information to help you differentiate between events that have `failed` versus those that are in `error`.

Some Workflow Events May Fail

Some workflow events may fail. For example:

- An event that validates an incoming notification may result in failure if the notification contains invalid information.
- An event that asks for the confirmation from a distribution company may fail if the distribution company rejects the request.

Fastpath: Refer to *Errors versus Failure* for information to help you differentiate between events that have `failed` versus those that are in `error`.

The background process that is responsible for activating events (referred to as `WFET`) is the process that can cause an event to fail (failure can happen during the activation algorithm on the workflow event). When an event fails, the system:

- Updates the workflow process with message number and message parameters describing the validation problem.
- Sets the status of the event to `Failed.`
- Cancels the workflow process and its outstanding events.
- Calls the *failure algorithm* defined on the event's event type.

It's important to note that some types of events can't fail and therefore don't have a failure algorithm.

Fastpath: Refer to *Workflow Processes Can Have Multiple Branches* for more information about event failure in a process with multiple branches.

Errors versus Failure

As explained under *Workflow Event Lifecycle*, an event's activation algorithm may cause a `pending` event to become in `error` or to `fail` (amongst other things). You control the exact state when you design your workflow event type activation algorithms.

The main differences between these two states is as follows:

- As described under *Some Workflow Events May Fail*, a failed event causes the entire workflow process to fail (and it cannot be restarted).
- As described under *Events May Be In Error*, a user can correct the cause of an error event's error and then change the event's status back to `pending`. The `pending` event will be processed the next time the activation process runs.

You should follow the following guidelines when designing your validation logic in your workflow event activation algorithms:

- If the cause of the problem is correctable by a user, you should set the state of the event to be in `error`.
- If the cause of the problem is not correctable by a user (e.g., you were interfaced information that cannot be corrected by your users), you should set the state of the event to `fail`.

Waiting Events And Their Waiting Process

Some events have to wait until something else happens before they can be `Completed` (or `Fail`). These types of events exist in the `Wait` state until the thing they are waiting for happens.

Every type of event that waits for something else to happen before it completes or fails must have a corresponding background process that monitors the thing on which the event is waiting. We refer to background processes that perform this monitoring as Waiting Processes.

There will be a Waiting Process for every type of event that has to wait for something to happen. The specific background process is defined on the workflow event type. For more information, refer to *Designing Workflow Event Types*.

The following points describe the responsibilities of a Waiting Process:

- Check on the thing on which the event is waiting. For example,
  - An event that creates a request to confirm a customer's request to switch suppliers has a Waiting Process that checks if the confirmation is accepted or rejected.
- Change the state of the event to `Complete` or `Fail` based on what transpired. For example,
  - The acceptance or rejection is received, the Waiting event can Complete or Fail
- Detect that the event has been waiting too long and do something. For example, the Waiting Process could:
  - Create a To Do entry (this might be useful if you need an operator to do something)
  - or, create an outgoing notification informing the sender of the incoming notification that something is wrong
  - or, `Fail`/ `Complete` the event (you may be able to automatically assume success or failure if an event waits longer than a predefined limit)
  - or, execute the event again and reset the base time on the event - if it waits too long, you could simply create another outgoing notification)
  - or, whatever else you can think of developing in the process

How Are Workflow Events Canceled?

The background process that is responsible for activating events (referred to as `WFET`) automatically `cancels` workflow events when an event `fails`.

A `pending` event will be cancelled automatically by the system if the workflow *process* is cancelled (a workflow process will be cancelled if an event fails or at the discretion of an operator).

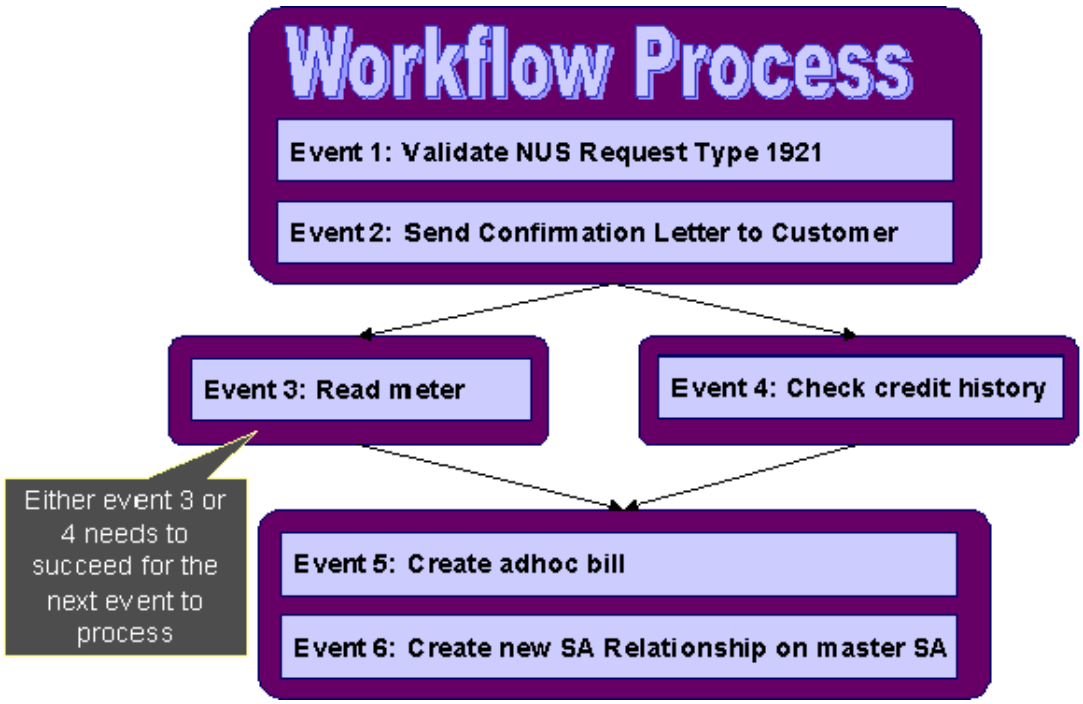
In addition, an operator may cancel a workflow event at will.

An event will be cancelled by the background process that is responsible for activating events (referred to as `WFET`) when it detects that ALL of its earlier, dependent events are cancelled. This is important to understand if your organization has *Workflow Processes With Multiple Branches*.

Workflow Processes Can Have Multiple Branches

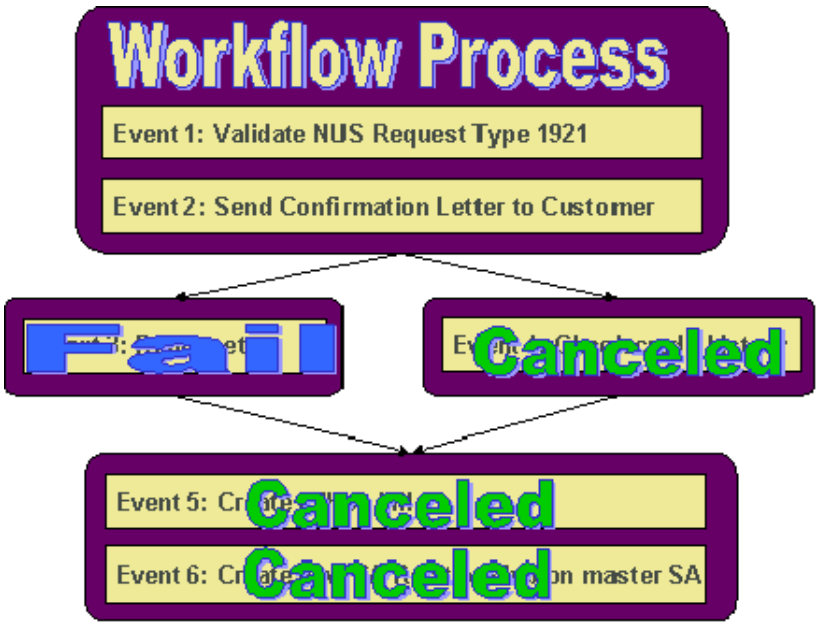
Using the workflow event dependencies, a workflow process may have multiple branches. There are several reasons why you may want to set up a process to contain multiple branches:

- There may be events that can run in parallel. This is useful if the related tasks take time to execute. The following is an example of such a workflow process.

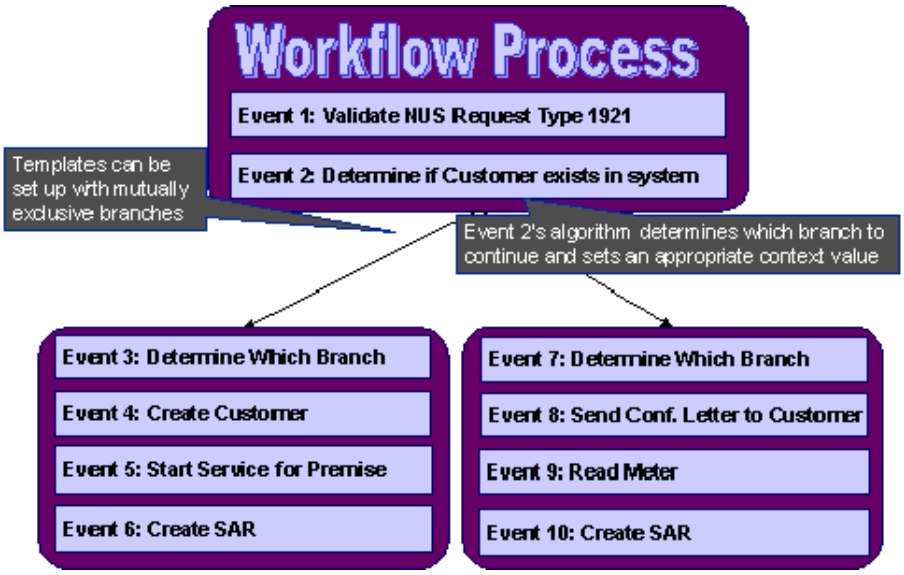


In the example above, if event 3 is canceled and event 4 completes, (or if event 4 is canceled and event 3 completes), event 5 will proceed. However, if both event 3 and event 4 are canceled, all remaining events will be canceled.

It should be noted that if either event 3 or 4 *fails*, ALL events in the process will be canceled, including events in a different branch.



- You may have a business process that has some common events and some events that are mutually exclusive. Rather than setting up several processes, you can set up one process that branches based on specific criteria.

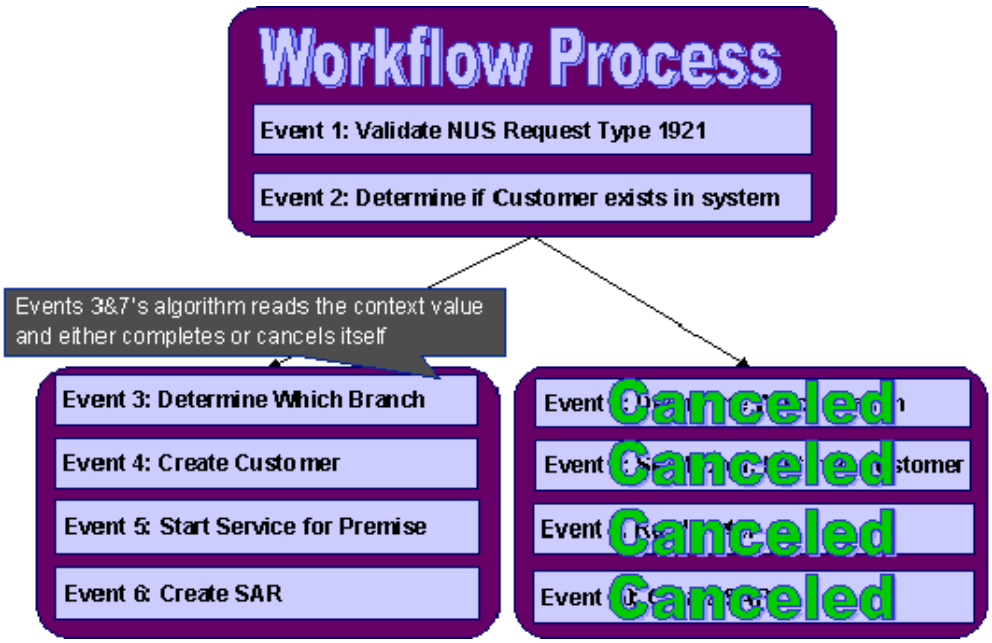


In the above example, event 2 determines if the workflow process is associated with an existing customer or a new customer.

- For a new customer; events 3, 4, 5 and 6 are executed.
- For an existing customer, events 7, 8, 9 and 10 are executed.

Event 2 does this by creating an entry in the Context collection indicating which branch should continue. (For example Context Type / Value of BRANCH/ A). Events 3 and 7 should be special events whose purpose is to read the BRANCH context type and determine if its branch should continue or be canceled.

In this example, Event 3's algorithm will continue if the context value for BRANCH is A and Event 7 will only continue if the context value for BRANCH is B.



Note: The above diagram and description indicates the use of context records to determine which branch to continue. If your implementation chooses to use characteristics instead, the same logic above may be implemented using characteristics as well.

The Big Picture of Notification Processing

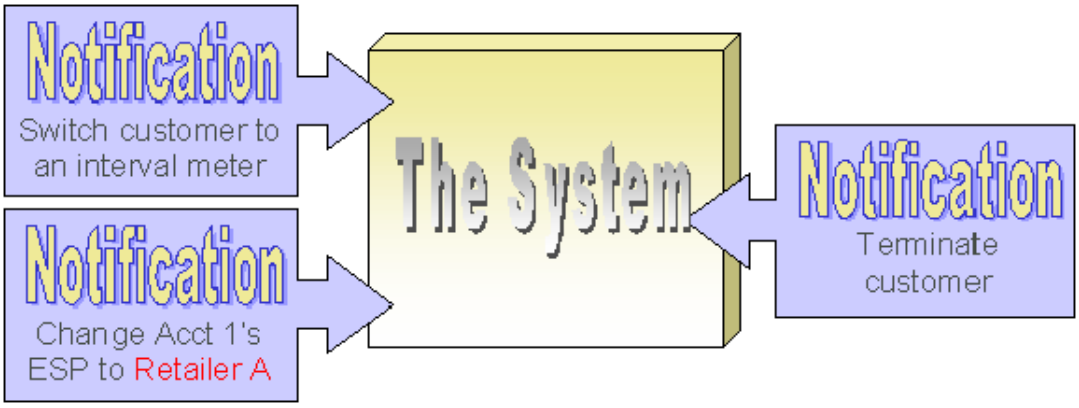
We use the term "notification" to reference the electronic transactions that you exchange with third parties when:

- They need information about a customer.
- They need to change something about a customer.

For example, an electric service provider may send a notification to an electric distribution company when a customer elects to use them as their energy provider.

You may have to support a wide variety of notifications. For example:

- You may receive a notification that tells you to switch a customer's energy service provider to a different provider (if you are a distribution company).
- You may receive a notification that asks for the last 24 months of consumption history for a customer (if you maintain consumption).
- Etc.



When a notification is received, the system responds by creating a workflow process. The related workflow process contains workflow events and it is the events that actually do whatever needs to be done (e.g., change the customer's service provider, stop service, etc.).

In addition to incoming notifications, the system creates outgoing notifications when:

- It needs to respond to an incoming notification. For example, if an incoming notification requests a customer's consumption history, the system must send the consumption history by creating an outgoing notification.
- It needs to apprise a third party that something has changed about the customer. For example, if a customer stops service, the system must tell the various service providers of such.

The topics in this section describe how notifications are interfaced into and out of the system.

Uploading Notifications Into The System

The following diagram illustrates the steps involved in the uploading of notifications into the system.

When a notification's electronic transaction is received, it must be inserted into the notification upload staging (NUS) table. Why? Because the system periodically looks for pending records in the NUS table and attempts to create a workflow process to implement the notification's request.

It's important to be aware that the events in a workflow process execute the notification request. To help solidify this concept, consider this - the first event in a workflow process typically validates the attributes on the NUS record.

Fastpath: Refer to *Notification Upload Background Process* for more information about the upload process.

What Type Of Workflow Process Is Created?

Given that you can receive many different types of notifications, it should make sense that each type of notification upload staging (NUS) record will probably require a different type of workflow process to implement its request. The system uses the NUS record sender's external system Id and a notification upload type to determine the type of workflow process. This works as follows:

- Every external system references a workflow process profile.
- A workflow process profile defines the type of workflow process template to use to process a given notification upload type.

Note: Different workflow processes for the same type of notification. It's probably obvious why different notification types result in different workflow processes. You may wonder why different workflow processes would be needed by two senders who submit the same type of notification? The reason is that different senders may have different types of computer systems that handle their notification processing (or no computer system at all). For example, you may have to respond by fax to a sender who doesn't have a sophisticated computer system, whereas you may send an electronic transaction to a sender who is technically advanced. The system allows you to create different workflow process profiles for each response mechanism (e.g., sophisticated vs. fax). You then associate the appropriate workflow process with each sender.

How Are Notifications Sent Out Of The System?

In addition to processing incoming notifications from third parties, the system also sends notifications to third parties to provide them with information (and to ask them for information). It is also possible to use notifications to send information to other applications within your company.

A Notification Download Staging (NDS) record is created for every notification that is sent to the outside world. NDS records are created by workflow events (i.e., the event's activation algorithm creates a NDS record). Consider the following examples:

- Many workflow processes exist to process *incoming* notifications. These processes typically contain workflow events that create NDS records to communicate with service providers. If you look at the workflow processes described under *Designing Workflow Process Templates*, you will see many such examples.
- When something noteworthy happens, the system may need to tell a third party about it. Or perhaps you have third party software, which contains information from the system and your company needs to keep the information in sync.

Fastpath: Refer to *System Conditions May Trigger Notification and Workflow* for more information about triggering notification download staging records from within the system.

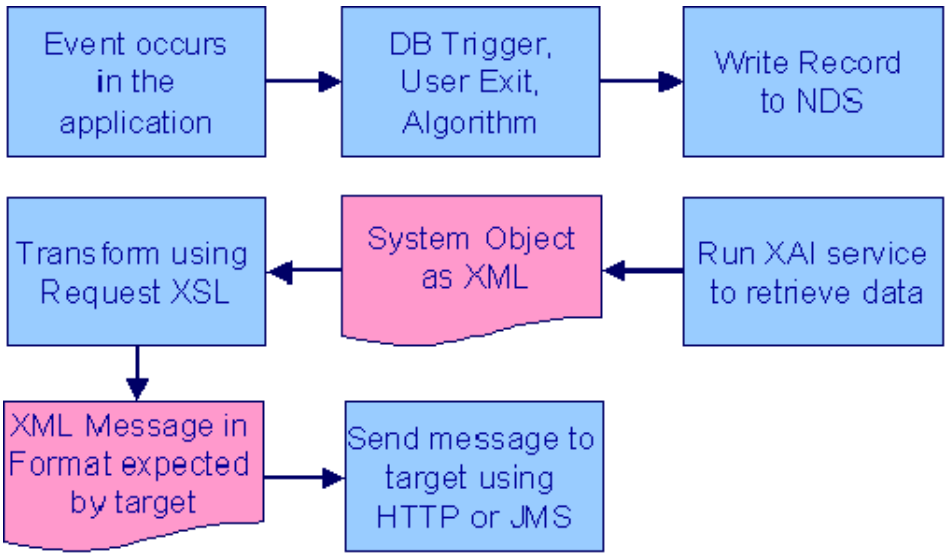
Notification and XAI

NDS records may also be processed using the XAI tool. Refer to Outgoing Messages for an overview of communicating outgoing messages via XAI.

The remaining sections describe the capability to send outgoing messages from the system to another application using the Notification and Workflow engine and XAI.

Near-Real Time Outgoing Messages

The following diagram illustrates the flow of near-real time messages:



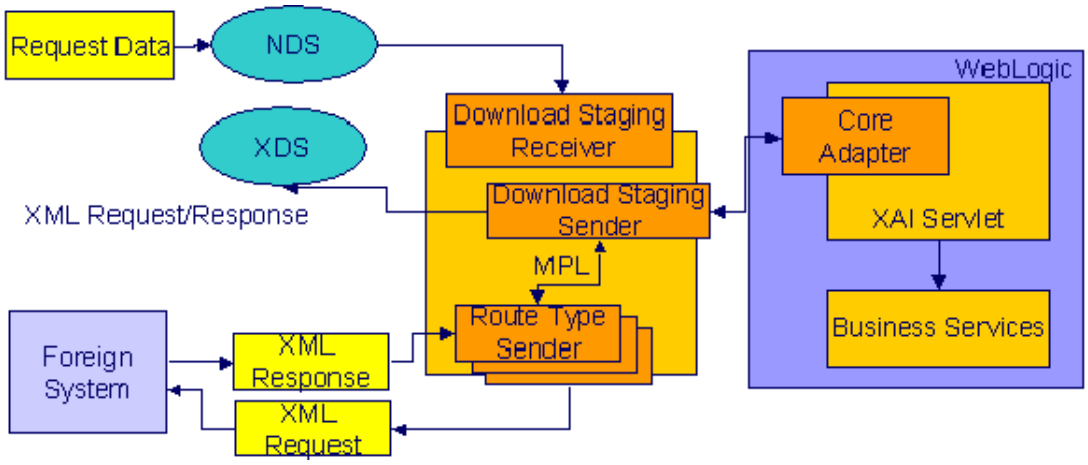
The following points describe the diagram

- When an event occurs in the system, you can initiate an outgoing message by storing a *notification download staging* (NDS) record. NDS records for outgoing messages have a special processing method flag of `XAI`.

Fastpath: Refer to *System Conditions May Trigger Notification and Workflow* for information about creating notification download staging records.

- Once a message is stored on the NDS table, the download staging receiver reads it and invokes XAI to extract data from your product. An XAI inbound service is used to retrieve the full information for the object related to the message.
- The download staging sender takes the response from the executer (the system object as XML) and continues the processing. It uses a request XSL on the route type to transform the message to a format expected by the target. It then sends the message to the target using the sender on the XAI route type.

The following diagram illustrates the process:



The responsibilities of the download staging receiver and download staging sender are provided in detail below.

Download Receiver

The download staging receiver processes records in the NDS table that have a processing method flag equal to `XAI` and a status of either `pending` or `retry`. It also uses the priority flag on the NDS type and the creation date/time on the NDS to process records according to their priority.

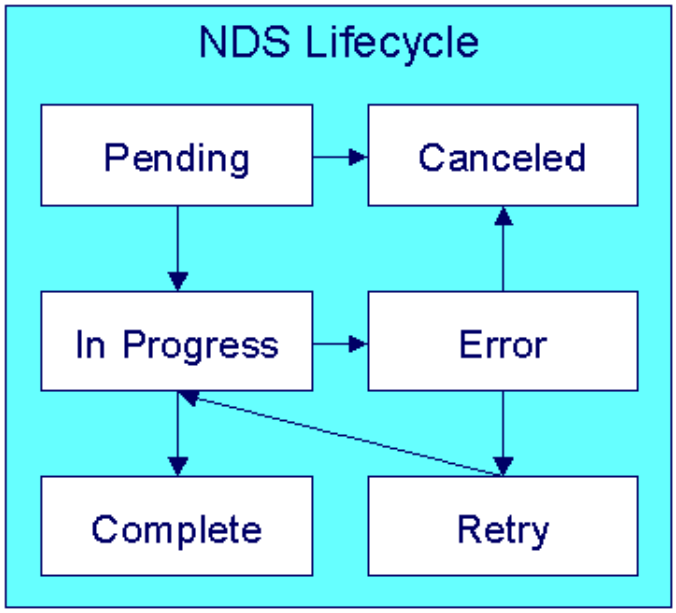
The process is referred to as "near real time" because your download staging receiver should be continuously checking for new records in the NDS table.

When it finds a record, it invokes XAI providing the XAI inbound service.

The following section provides more information about the NDS lifecycle.

Lifecycle of Notification Download Staging

The download receiver processes NDS records based on their status. The following diagram describes the lifecycle of an `XAI` type NDS.



The following points describe the diagram:

- Records are created in `pending` status. A user may transition the record to the `canceled` state.
- The MPL processes records in `pending` and `retry` status. While the record is being processed, it is changed to `in progress`.
- Based on the results of the processing, records could be transitioned to `error` or `complete`.

  Note: An NDS is set to error if there is a problem with the NDS or if any of its related XDS records are in error.
- A user may cancel a record in the `error` state or change the status to `retry` to have the MPL try again.

Building the XML Request

To build the XML request for the outgoing message, the download receiver needs the following information:

- The XML request and response schemas used to retrieve information about the record related to the request. For example, if the outgoing message is to inform an external system that a person's name has changed, the XML request schema must retrieve the appropriate person record.
- The *NDS type* related to the NDS defines an XAI Inbound Service that references an appropriate XML request schema.

- The data related to the request. In this example, the NDS record should be stored with the ID of the person record that changed.
- The person ID is stored as NDS context, with an appropriate Context Type, like `PERID`.
- XPATH information to tell the system where to plug in the related data into the request schema. In this example, when building the XML request, the person id must be plugged into the appropriate location in the XML request schema prior to executing the request.
- The *NDS type* related to this NDS defines a collection of context types. For each context type, you define an *XPATH* used to indicate where to substitute the context data.

Once the request schema is build, the receiver invokes XAI (via the executer) to execute the XML request. In our example, it calls the appropriate Person service and the XML response includes the extracted person information.

Download Staging Sender

This piece is somewhat confusing, because the download staging sender behaves differently than other senders. *Sender* are typically responsible for

- Routing a request to an appropriate target
- In the case of the upload staging sender and the staging control sender, the sender is responsible for updating the status of the appropriate table.

In the case of the download staging sender, it's responsible for processing the "response" to the XAI executer, whose task was to build the XML request.

- If there is a problem executing the request, the download staging sender updates the NDS record in `error` and creates an *NDS exception* record.
- If the executer was successful, the "response" is an XML request in a format understood by your product. The sender must perform the following steps to help process the outgoing message:
  - Transform the request to a format expected by the target system(s)
  - Route the outgoing message to the target
  - Create an XAI download staging record for each message sent out. (There is typically only one, but it is possible for the outgoing message to have *multiple destinations*.)

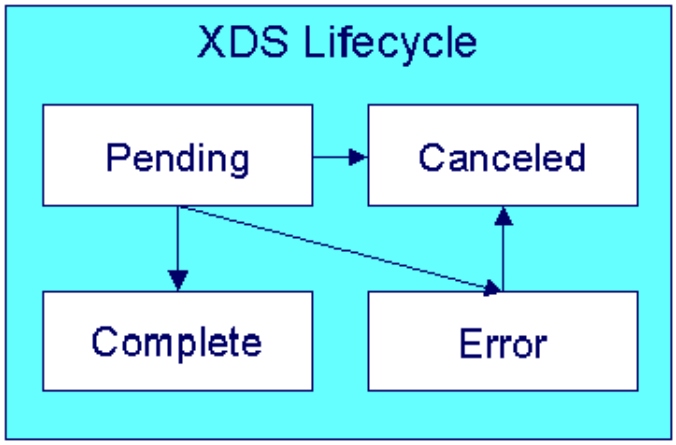
If any errors are found during any of these steps, the status of the NDS record is set to `error` and a record is written to the *NDS exception* table. If the error is related to routing the message, the XDS is created in `error` and a record is written to the XDS exception table.

> **Note: Routing Optional**. It is possible that an *NDS message does not require routing* information. In this case the download staging sender simply updates the NDS record's status

The following sections provide further detail.

Lifecycle of XAI Download Staging

The following diagram illustrates the lifecycle of an XAI download staging record.



An XAI download staging record is created in one of the following ways:

- When the download staging sender continues the processing for the outgoing message, it attempts to transform and route a message for each XAI route type indicated on the notification download profile. An XDS is created to record the message sent.
  - If the message was sent successfully, the XDS is created in `complete` status.
  - If an error occurred when sending the message, the XDS is created in `error` status and an XAI download exception is created.
- When the system attempts to send a message *real time* but the external system is unavailable, the system may create a `pending` NDS and a `pending` XDS record.
- A user may `cancel` a `pending` or error XDS record.

If you have resolved the error for an XAI download staging record, change the status of the related NDS record to retry. When MPL processes the NDS again, it deletes all XDS records in error and attempts to send them again.

Transform the Request

The download staging sender takes the XML response with the extracted data and attempts to transform into a format accepted by the target. It needs an appropriate XSL transformation script.

The XSLT is defined on an *XAI Route Type*, which is referenced on the *download profile* of the service provider referenced by the notification download staging record.

Routing the Message

In addition to defining the appropriate XSL transformation scripts, the XAI route type also references a *sender* that tells the system how to route the message, for example it may be routed using a JMS queue.

If the target supports synchronous communication (for example, an HTTP sender), the *route type* may be configured to receive an acknowledgement. This indicates to the download staging sender to wait for a response from the sender on the route type.

The download staging sender creates an appropriate XDS record with the XML request document and with a status of `complete` or `error` based on the results of the communication with the target. If the target has sent a synchronous response, the XML response is also posted on the XDS.

If the target supports synchronous communication, you may indicate that a response to the outgoing message also requires action. For example, perhaps the outgoing message to an external system causes a new record to be created in that system. The response to the message may require an update to our initiating record to post the external system's identifier for the record on their side.

To enable this logic, you must check the Post Response switch on the *route type*. If this has been checked, the download staging sender creates an XAI upload staging record (and a staging control record) and it links the XDS record as a foreign key. The XUS record is processed along with other uploaded messages, to invoke an appropriate service.

> **Note: XML Response**. If the Post Response switch on the route type is turned on, the XML response that is captured is the XML that results after applying the Response XSL. If the Post Response switch is not turned on (i.e., the response is a simple acknowledgement), XML response is posted without applying a Response XSL. (In fact, for simple acknowledgements, there is no need for a Response XSL.)

> **Note: Asynchronous Responses**. Refer to *Asynchronous Responses to NDS Messages* for information about processing responses to messages sent asynchronously.

Multiple Message Destinations

If your message must be sent to more than one external system, a different XAI route type is required for each system that the message must be sent to so that the sender and XSLT for each external system can be defined.

The recommended procedure for sending a message to multiple destinations is to

- Generate an NDS for each destination where each NDS defines the same NDS type but different service providers. In other words, the service provider represents the destination.
- Each service provider references a *notification download profile* where the appropriate XAI route type is defined for each NDS type.
- When this NDS record is processed, the sender information defined for the XAI route type is used to route the message appropriately and an XAI download staging (XDS) record is created.

Note that it is possible for the download profile's formatting method to reference more than one XAI route type. This is perhaps another configuration that could be used to send a message to multiple destinations. In this case an XDS record is created for each XAI route type to track the XML request. However, it is not possible to track *asynchronous responses* to multiple messages using this mechanism. Defining multiple route types for a download profile formatting method should only be done if you do not expect an asynchronous response.

An NDS Message That Doesn't Require Routing

If an NDS message is being sent to one of your own external systems and this system may be accessed directly, you could design your message such that XAI routing is not necessary. For this scenario, the assumption is that the application service that is invoked by the XAI inbound service completes all the necessary steps to update the external system.

If this is the case, you are not required to define an XAI Route Type on the appropriate notification download profile entry. In addition, no XAI sender is required for this external system.

The download staging sender's sole responsibility for a message of this type is to update the NDS status.

Asynchronous Responses to NDS Messages

If you send an asynchronous NDS message and you expect a response, the response is received as an inbound message. In order to recognize that an inbound message is a response to an NDS message, there must be some handshaking. In other words our system must pass a unique identifier of our message to the external system and the external system must include that identifier in the response.

Populating the Unique Message Identifier

As described in *Building the XML Request*, the information that may be included in the XML request we build is information linked to the NDS record as context. Therefore, the unique identifier you want to send to the external system must be stored as a context record. (Let's call this context type Message ID). Because the notification download staging ID is a unique identifier, you may choose to populate Message ID with the NDS ID. However, it is possible that the external system requires an identifier in a different format. For example, the NDS ID is a 12-byte number. Perhaps the external system can only receive a 10-byte number for the Message ID.

It is the responsibility of the mechanism that creates the NDS record (i.e., the algorithm or user exit or database trigger) to create a context entry for Message ID with an appropriate value that is compatible with the target system.

When creating your XML request schema, if you expect a response to your message, you must include the message ID as an attribute in the XML request. Because the message id is not part of the product information that your request schema is extracting, it should be defined as a private attribute. Once the XML is created, it is the responsibility of the XSL to transform the Message ID into the target specific XML element.

Note: Only One Route Type. Because the unique Message ID for an NDS message is linked to the NDS record via context, it is not possible for the system to handle multiple route types (and therefore multiple senders) for a single message IF an asynchronous response is expected. The reason is because all senders receive the same unique Message ID and if each sender responds, the system is not able to determine which response is received for which sender.

Recognizing the Inbound Message Response

When the external system sends an asynchronous response to your NDS message, it is received as an inbound message. This inbound message must include the Message ID that you sent so that you can recognize this as a response. In addition, the XSL transformation must populate the identifier of the external system sending the message.

The following points describe the detail related to receiving an inbound message:

- When a response comes in, the message is translated using an XSLT script. The XSLT must map the message ID, which is the unique identifier of the NDS message that this is a response to, to the XML element `MessageID`. It must create an attribute with the XML element `ReplyToMessageSource` populating the value with an appropriate *external system*. This value should correspond to the external system code of the service provider related to the NDS message.
- The XML request of the inbound message is processed as normal.
- The existence of the special message elements in the XSL indicates to XAI that XML response processing is required. It looks for a `complete` NDS with a service provider whose external id matches the `ReplyToMessageSource` and with a context type of `Message ID` whose context value matches `XAI Message ID`.
- When the correct NDS is found, the system finds the related XAI download staging record and populates the XML response with the XML request of the incoming message.
- If the XDS request is not associated already with a response or the latest associated response is in `error` status stamp the current response on the XDS request record.
- If the response came into the system as an XAI upload staging (XUS) record, the foreign key to the XDS record (NDS ID and XAI route type) to which it is responding is linked to the XUS.
- If the response did not come in as an XAI upload staging record (for example, it came in via JMS queue), an XAI upload staging record and an XAI staging control record are created in `complete` status as an audit and the foreign key to the XDS record is linked to the XUS.

Note: Errors Linking the XDS. If the system has a problem finding the appropriate XDS record to link to the incoming response, the inbound message is still processed. A log entry is added to the XAI trace file reporting the error in identifying the correct XDS record.

Real Time Outgoing Messages

This section describes the capability to send real time messages from the system to another application. The functionality works in conjunction with the near real time NDS message configuration.

Real Time Message Engine

The system provides an "engine" to communicate real time messages to an external system. This engine may be invoked by a user exit on any user interface in the system. The real time message engine receives an NDS type and a service provider as input. Using this information, the engine finds the appropriate XAI route type for the service provider's notification download profile. The route type indicates the XSLT and the XAI sender.

Note: HTTP Sender. In the current release of the product, only senders that communicate via HTTP are supported.

Unlike near real time messages, the XAI inbound service referenced on the NDS type is not used to build the XML request. Rather, the user exit that invokes the engine is responsible for extracting the appropriate data and passing to the engine a formatted XML document. The XSLT referenced on the route type should map attributes from XML document into a format expected by the target system.

The following points describe how the real time engine works:

- Once the XML request is built, the message is sent out based on the XAI route type's sender information.
- The system waits for a response for a predefined time out limit defined as a context entry on the *sender*.
- When the response is received, the engine transforms the message using the response XSL defined on the route type.
- The XML response is then passed back to the user exit that called the engine. The user exit processes the response as needed.

Note: **Note**, as described in Routing the Message, the route type includes a flag to indicate to the system if a response requires additional action. If so, an XUS is created. The real time message engine does not support this logic. It expects the user exit to process the response as needed.

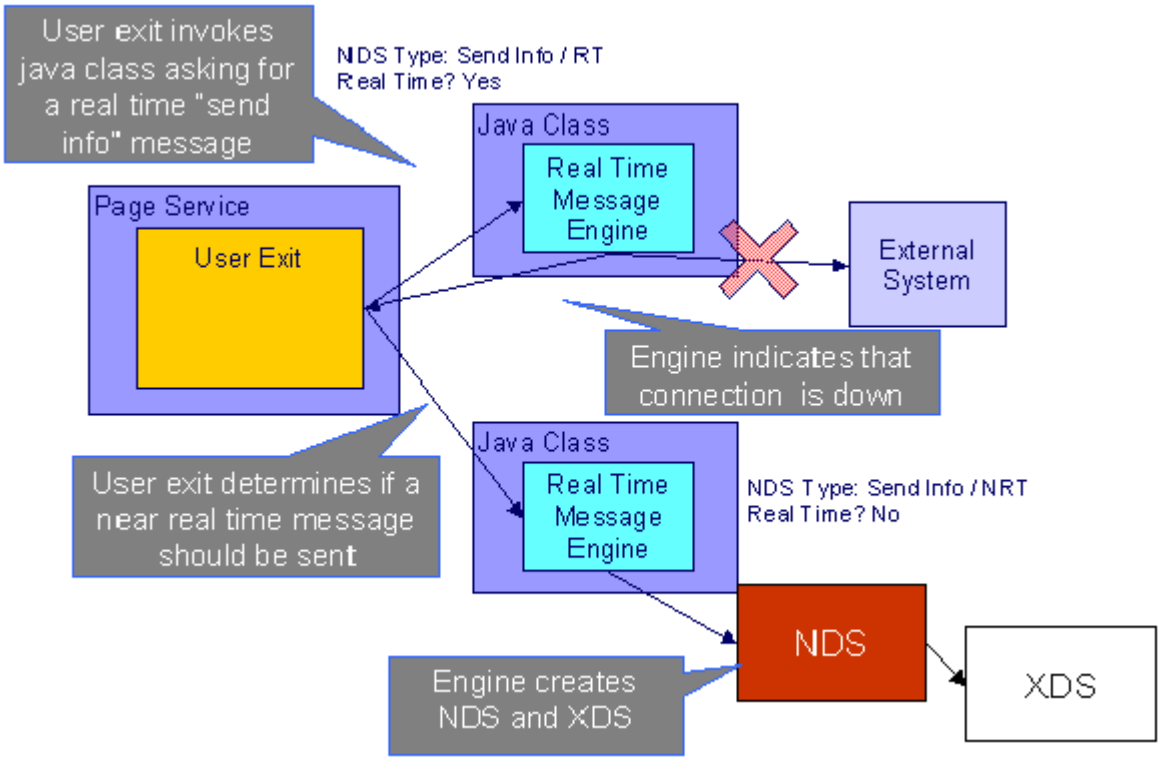
Sending Real Time Messages in Near Real Time

If the connection is not available when the real time engine attempts to communicate with the target, the engine may post the message to the download staging tables to send in "near real time" when the target system is available again.

Messages should only be posted to the staging table if the user does not need to wait for a response in real time. Consider the following examples:

- A real time message is used to request information from an external system to display to a user on a page. If the connection is down, it does not make sense to store a near real time message for this case because the user is not able to wait for the response to the request.
- There is a connection problem when the user attempts to send information to an external system. In this case, perhaps it makes sense to store a near real time message so that the information is sent to the external system when the connection is restored.

Note. The real time message engine does not make the decision to post a message near real time when the communication is unavailable. The responsibility lives solely with the user exit that calls the java class that invokes the engine. The following diagram illustrates a possible interaction between the user exit and the message engine:



The following points describe the diagram:

- When the user exit is informed that the real time message could not be sent, it is the responsibility of the user exit to decide if the engine should be invoked again to create a near real time message.
- The NDS type used for near real time messages is different than the one used for real time messages. The reason is because the XSL transformation script is different for real time vs. near real time.
- The NDS is created and marked `pending`.
- The XDS record is created and marked `pending`. The XML message that is ready to be sent to the sender is posted as the XML request for the XDS.

Because the XML request is already built, the NDS type used for the NDS should reference a special XAI inbound service. This service basically tells the download staging sender that the XDS already exists and doesn't need to be created.

System Conditions May Trigger Notification and Workflow

Note: Outbound Messages. The mechanism for triggering communication to an external system described here does not take advantage of the configurable business object functionality that enables outbound messages to be triggered from a server-based script.

There may be many noteworthy occurrences in the system, which should trigger the notification and workflow process.

Some of these occurrences may need to be communicated to an external system. Examples of when this may occur are as follows:

- You use an external software package, which uses information from the system and you need to send a message to this software package when certain attributes of the customer change.
- You need to inform third parties when something changes about a customer. For example, if a customer, who has been referred to a collection agency, makes a payment, you will need to inform the agency of the payment.

In many scenarios, you may want more than just a notification to be sent out, but you would also require a workflow process. For example, perhaps when a customer's mailing address changes, you want to a) send a confirmation letter to the customer's old address and their new address; b) inform a third party service provider about the change; and c) send the update to an external marketing system. In this case, the act of changing the mailing address could trigger a workflow process, which would perform the above steps.

You may have other scenarios where a noteworthy condition in the system should trigger a workflow process, which does not require a notification to be sent out of the system. For example, perhaps your company follows a sophisticated procedure for starting service, which requires certain events to transpire before other events can occur. You could create a workflow process, which gets triggered upon starting service to handle this sophistication.

In summary, when something occurs in the system, you may need to

- Create a notification download staging (NDS) record. You would do this if you only need to send a message to an external system.
- Create a notification upload staging (NUS) record. You would do this if you need to trigger a workflow process to do one or more steps. One of the workflow events in this process could create a NDS if an external system needs to be informed.

Creating a Notification Download Staging Record

To create an NDS record directly, you need the following information:

- Service Provider (who will receive the outbound message)
- NDS Type

You also need to populate relevant data using either context or characteristics. For example, if a change to a person's name should trigger an outbound message, the Person ID must be referenced.

The process that creates the NDS must determine the appropriate service provider that should receive this information based on the business rules. As for the NDS type, this is typically "hard-coded" based on what has occurred to trigger the outbound message. For example, if the customer's name changed, then the NDS Type is set to something like "Name Change".

For any NDS records automatically created by the system in sample algorithms or in system processes, the NDS type is not hard-coded. Rather, the system uses a download type condition flag value to look up the appropriate NDS type. As a result, if an implementation chooses to use these sample algorithms, but prefers to use a different NDS type, it is possible. When designing your own NDS types that are referenced by NDS records created based on system conditions, you may choose to create new download type condition flag values as well.

Refer to *Designing Notification Download Types* and *Setting Up Notification Download Types* for more information.

Creating a Notification Upload Staging Record

To create an NUS record directly, you need the following information:

- External System (to point to a service provider)
- NUS Type

You also need to populate relevant data using either context or characteristics. For example, if a change to a person's name should trigger a letter to the customer and an outbound message, the Person ID must be referenced.

The process that creates the NUS must determine the appropriate external system based on the business rules. If the steps that need to be taken as a result of this system condition are based on an internal practice, the external system could be something like "SYSTEM". That external system would reference an appropriate workflow process profile that defines the appropriate workflow process based on the NUS type.

Determining the NUS type is similar to determining the NDS type as described in the previous section.

Refer to *Designing Notification Upload Types* and *Setting Up Notification Upload Types* for more information.

Triggering Notification & Workflow from the Application

You and your implementation team must determine the noteworthy events that should trigger workflow and/or outgoing notifications based on your business practice.

Next, you must assess how you are going to create the appropriate records:

- Determine if there is a plug-in spot that exists where you can create an algorithm to trigger the NUS or NDS. For example, if canceling a contract should trigger a notification, an Contract Cancel algorithm plugged in to the contract type may be used.
- If there is no plug-in spot, determine if there is a user exit at the appropriate place for you to trigger the creation of an NUS or NDS
- You may also decide to use a database trigger to create the NUS or NDS

Once you have determined how and when to trigger the creation of notification records, have your implementation team create the CM code to do the work.

The Lifecycle of Notification Download Staging

If a download staging record is routed to the outside world via XAI, a status is assigned to the download staging record.

Refer to *Lifecycle of Notification Download Staging* for more information.

Creating Notification and Workflow Procedures

A workflow process is created by copying the events defined on a workflow process template (a workflow process template contains the standard events).

Note: Workflow processes can be used for any multi-event process. While this section describes how to design workflow processes to support incoming and outgoing notifications, workflow processes can be used for other multi-step processes.

The topics in this section describe how to design and setup the tables that control your notification and workflow processing.

Fastpath: For more information about notification and workflow processing, see *The Big Picture Of Workflow Events* and *The Big Picture Of Notification Processing*.

Caution: There are innumerable ways to design your notification and workflow procedures. Some designs will result in easy long-term maintenance, others will result in maintenance headaches. In this section, we provide information to help you understand the ramifications of the various options. Before you set up your production procedures, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

Designing Notification Upload & Workflow Procedures

The topics in this section describe how to design the tables that control your notification upload and workflow processing.

Designing Workflow Process Profiles

The following sections describe how to design a workflow process profile. A workflow process profile controls the type of workflow process that will be created to process your incoming notifications. If you plan to use workflow processes outside of the notification upload process, you can skip to *Designing Workflow Process Templates*.

You associate a workflow process profile with one or more external system. Whenever a notification is received from an external system, the system uses the workflow process profile to determine the type of workflow process to create to implement the notification.

The easiest way to design a workflow process profile is to choose a representative external system and design a workflow process profile for it by filling in the matrices below. After you've designed a profile, determine how many other external systems can use it. Then design the next external system's profile and determine who can reuse it. Repeat this process until all your external systems have a profile. Once the profiles are designed, you're ready to set up the workflow control tables.

Fastpath: Refer to *The Big Picture Of Notification Processing* for more informationabout how workflow processes are used to implement incoming notifications.

The topics discussed below will gradually complete the following matrix using a simple case-study. We recommend that you use the following matrix as your guide. When the matrix is complete, you're ready to set up a workflow process profile.

WF Process Criteria **Notif. Upload Type**		

Designing Notification Upload Types

You will need one notification upload type for every type of notification your organization can receive from the various service providers. To "design" your notification upload types, you document the codes used by external systems to identify the transaction types on their notification upload staging records.

We have populated the Notification Upload Type column with a few classic examples that can be received by a distribution company:

WF Process Criteria **Notif. Upload Type**		
Retrieve a customer's consumption history		
Switch a customer to a different service supplier		

Note: It is recommended that the unique identifier of your Notification Upload Types match the "transaction types" that are referenced on incoming notifications. If it is not possible to do this, the process that creates the upload staging records must map the external transaction types to the notification upload types that you define in the system.

In addition, as described in *System Conditions May Trigger Notification and Workflow*, you should evaluate any condition that should cause an NUS to be created (to eventually create a workflow). A new NUS Type should be defined for each condition. For each of these, consider creating a new value for the upload condition flag so that the NUS type is not hard-coded in the system process that creates the NUS.

Navigating to Related NUS Extension

When designing the types of notifications your organization may receive, you must determine where you plan to store the detailed information related to the incoming transaction: as context linked to the NUS, as characteristics linked to the NUS or by using an extension record.

Fastpath: Refer to *Process X - Populate Notification Upload Staging* for information about the different options available.

If you choose to capture the detailed information in a new extension record with a new user interface, then in order to be able to drill down to the related extension record, the following steps are required.

- Add a new *lookup* value for your new extension record for the field `NT_UP_EXTSN_FLG`. (This step is needed regardless of whether you want to drill down to the related record.)
- Add a *navigation option* pointing to the appropriate navigation key for your new transaction. This navigation option must reference a usage type of `notification upload type`.
- Indicate the appropriate lookup value and navigation option on the appropriate notification upload type.

Designing Workflow Process Criteria

The matrix's second dimension is dependent on workflow criteria algorithms. Workflow criteria are confusing. Think of them as optional conditions that, if met, will cause a different type of workflow process to be started when a given notification upload type is received.

You must define a `Default` criterion in case none of the override criteria are met. You MAY have override criteria if different situations result in different types of workflow processes. For example, let's assume some notification upload types have different workflow processes for industrial customers as compared to all other types of customer. This assumption necessitates the introduction of an override workflow process criteria; we'll call it `Industrial Customer`.

WF Process Criteria **Notif. Upload Type**	**Default**	**Industrial Customer**
Retrieve a customer's service history		
Switch a customer to a different service supplier		

Note: The workflow process criteria are limited by your imagination (and business requirements). We have provided the workflow process criteria you see above as an example; we don't expect you'll be able to use the exact conditions we supply. Your conditions will be based on any number of factors.

Note: New workflow process criteria may require programming. See *How To Add A New Algorithm* for more information.

Designing Workflow Processes To Process Incoming Notifications

The next step involves populating each cell in the matrix with the workflow events that should be executed when the system receives a notification identified with a given notification upload type. If override criteria aren't relevant for a given notification upload type, we will mark the cell as "N/A".

WF Process Criteria **Notif. Upload Type**	**Default**	**Industrial Customer**
Retrieve a customer's consumption history	Validate notification - consumption history request Confirm requester is a valid service provider for the customer's service. Create notification download - send consumption history	N/A (meaning that industrial customers use the Default criteria)
Switch a customer to a different service supplier	Validate notification - supplier switch Confirm requester is a valid service provider for the customer's service. Check with current supplier if the switch is allowed. Switch suppliers.	N/A (meaning that industrial customers use the Default criteria)

Note: Notice that the first event in each cell typically validates the notification upload staging record.

At this point, the matrix is complete. Before you're ready to design your workflow process templates you must design the processes described in the next section.

Designing Workflow Processes To Deal With Invalid Sender or Notification Upload Type

The system needs to know the type of workflow process to create when a notification is uploaded that does not contain a valid External System or Notification Upload Type (these two fields are the ones that control the type of workflow process that's created to process the uploaded notification). When these conditions are detected by the notification upload process, most financial services create an outgoing notification rejecting the uploaded notification when such conditions transpire. We've shown these in the following table.

Notification Condition	Workflow Process Events
Unknown Notification ID	Create notification download - reject notification, bad external system
Unknown Notification (Upload) Type	Create notification download - reject notification, bad notification upload type

Note: These processes are not in a workflow process profile. The above workflow process templates are not referenced in a workflow process profile. Rather, when you create these workflow process templates you label them with a **Notification Condition** of `Unknown Notification ID` or `Unknown Notification Type`. The notification upload process will then create workflow process when these events transpire.

Designing Workflow Process Templates

The following table shows the workflow process templates referenced in the previous section's matrix. Adjacent to each process is its events and an indication of when they are triggered.

Workflow Process Template	Event Number	Workflow Event Type	Dependent On Event(s)	Trigger Date Set To X Calendar Days After Completion Of Dependent Events
Retrieve a customer's consumption	10	Validate notification - consumption history request	N/A - first event	0
	20	Confirm requester is a valid service provider for the customer's service	10	0
	30	Create notification download - send consumption history	20	0
Switch a customer to a different service supplier	10	Validate notification - supplier switch	N/A - first event	0
	20	Confirm requester is a valid service provider for the customer's service	10	0
	30	Check with current supplier if the switch is allowed	20	0
	40	Switch suppliers	30	0
Reject bad notification upload type	10	Create notification download - reject request	N/A - first event	0
Reject bad external system	10	Create notification download - reject request	N/A - first event	0

Note: The workflow process for "Reject bad notification upload type" should reference the Notification Condition `Unknown Notification Type`. The workflow process for "Reject bad external system" should reference the Notification Condition `Unknown Notification ID`.

Designing Workflow Event Types

If we extract each unique event type from the above table, we end up with the following:

Workflow Event Type
Validate notification - consumption history request
Confirm requester is a valid service provider for the customer's service
Create notification download - send consumption history
Validate notification - supplier switch
Current supplier confirmation
Change rate
Create notification download - reject request

Next, we have to determine the algorithm that will be used when each event is activated on its trigger date. We call this algorithm the *activation algorithm*. An activation algorithm is a stand-alone routine that does whatever you need done when an event is activated. We have populated the following table with brief descriptions of the types of activation algorithms you'd need for the above workflow events.

Note: The activation algorithms are limited by your imagination (and business requirements). We have provided the activation algorithms you see below as an example; we don't expect you'll be able to use the exact algorithms that we supply. Your algorithms will be based on any number of factors. Be aware that new activation algorithms may require programming. See *How To Add A New Algorithm* for more information.

Workflow Event Type	**Activation Algorithm**
Validate notification - consumption history request	Validate consumption history request
Confirm requester is a valid service provider for the customer's service	Confirm service provider is valid requester
Create notification download - send consumption history	Create notification download - send consumption history
Validate notification - supplier switch	Validate supplier switch request
Current supplier confirmation	Create notification download - check if it's OK to switch customer from current supplier
Switch supplier	Switch supplier
Change rate	Change rate
Create notification download - reject request	Create notification download - reject request

Next, we have to determine which types of events can fail. Refer to *Some Workflow Events May Fail* for background information failure. For those types of events that can fail, we will indicate their *failure algorithm* in the table. A failure algorithm is a stand-alone routine that does whatever you need none when an event fails. We have populated the following table with brief descriptions of the types of failure algorithms you'd need for the above workflow events.

Note: The failure algorithms are limited by your imagination (and business requirements). We have provided the failure algorithms you see below as an example; we don't expect you'll be able to use the exact algorithms that we supply. Your algorithms will be based on any number of factors. Be aware that new failure algorithms may require programming. See *How To Add A New Algorithm* for more information.

Workflow Event Type	Activation Algorithm	Failure Algorithm
Validate notification - consumption history request	Validate consumption history request	Create notification download - invalid request
Confirm requester is a valid service provider for the customer's service	Confirm service provider is valid requester	Create notification download - invalid requester
Create notification download - send consumption history	Create notification download - send consumption history	N/A
Validate notification - supplier switch	Validate supplier switch request	Create notification download - invalid request
Current supplier confirmation	Create notification download - check if it's OK to switch customer from current supplier	Create notification download - reject request due to supplier rejection
Change rate	Change rate	N/A
Create notification download - reject request	Create notification download - reject request	N/A

And finally, for those events whose activation algorithm puts them into a wait state, we have to determine the waiting process that monitors the waiting events. Refer to *Waiting Events And Their Waiting Process* for more information about waiting.

Note: The waiting processes are limited by your imagination (and business requirements). We have provided the waiting processes you see below as an example; you may not be able to use the exact processes that we supply as your processes will be based on any number of factors. Be aware that a new waiting process will require programming.

Workflow Event Type	Activation Algorithm	Failure Algorithm	Waiting Process
Validate notification - consumption history request	Validate consumption history request	Create notification download - invalid request	N/A
Confirm requester is a valid service provider for the customer's service	Confirm service provider is valid requester	Create notification download - invalid requester	N/A
Create notification download - send consumption history	Create notification download - send consumption history	N/A	N/A
Validate notification - supplier switch	Validate supplier switch request	Create notification download - invalid request	N/A

Workflow Event Type	Activation Algorithm	Failure Algorithm	Waiting Process
Current supplier confirmation	Create notification download - check if it's OK to switch customer from current supplier	Create notification download - reject request due to supplier rejection	Check if supplier has accepted process
Change rate	Change rate	N/A	N/A
Create notification download - reject request	Create notification download - reject request	N/A	N/A

Designing External Systems

As described under *What Type Of Workflow Process Is Created?*, every notification upload staging (NUS) record contains an external system. An external system is the unique identifier of the sender of the notification.

To "design" your external systems, document the external id's (e.g., DUNS number) of your service providers.

Note: It's obvious, but worth stressing, that the External System should be the exact number referenced on notifications sent by a sender.

Designing Notification Downloads

The system creates outgoing notifications when:

- It needs to respond to an incoming notification. For example, if an incoming notification requests a customer's consumption history, the system must send the consumption history by creating an outgoing notification.
- It needs to apprise a third party that something has changed about the customer. For example, if a customer stops service, the system must tell the various service providers of such.
- It needs to exchange information with another application in the company, for example a CRM system.

A Notification Download Staging (NDS) record is created for every notification that is sent to the outside world.

The topics in this section describe how to design the tables that control your notification download processing.

Designing Workflow Processes To Support Outgoing Notifications

As described under *How Are Notifications Sent Out Of The System?* notifications are sent out via Notification Download Staging (NDS) records. NDS records are created by workflow events (i.e., the event's activation algorithm creates notification download records). There are two types of workflow processes that contain these types of workflow events:

- Many workflow processes exist to process *incoming* notifications. These processes typically contain workflow events that create NDS records to communicate with service providers. If you look at the workflow processes described under *Designing Workflow Process Templates*, you will see many such examples.
- When something noteworthy happens, the system may need to tell a third party about it. As described in *System Conditions May Trigger Notification and Workflow*, you may decide that the noteworthy condition should cause a workflow process to be created where one or more of the workflow events can create an NDS. The following table shows representative workflow process templates of this type.

Workflow Process Template	Event Number	Workflow Event Type	Dependent On Event(s)	Trigger Date Set To X Days After Completion Of Dependent Events
Stop service	10	Create notification download - inform all parties that customer is stopping service	N/A - first event	0

After documenting these types of workflow process templates, follow the instructions under *Designing Workflow Event Types* to add any new types of events to the list.

Designing Your External System Feature Configuration

For certain types of external systems, your application may define options used to configure the interaction between your product and your external system. The feature configuration table is used to define some settings applicable to the interaction with your external system, for example:

- The Options grid allows you to define information needed to communicate with the external system.
- The Messages collection allows you to map each message that you may receive from the external system to a corresponding *message* in this system.

Note: Recommendation. We recommend creating a new message for every message that is being mapped. This ensures that changes to future base product messages do not affect the integration.

Define the XAI Sender

If any of your messages are routed to an external system via XAI, the message must also be associated with a *XAI Sender*, which tells the system how to send the message.

Note: Sender Optional. If the NDS message does not require routing, you are not required to define an XAI sender.

Designing Notification Download Types

You will need one notification download type for every type of notification your organization can send to service providers. To "design" your notification download types, list:

- Every outgoing notification that you documented under *Designing Workflow Event Types*.
- Every outgoing notification that you documented under *Designing Workflow Processes To Support Outgoing Notifications*.
- Every outgoing notification that may be triggered from within the system as described in *System Conditions May Trigger Notification and Workflow*. For each of these, consider creating a new value for the download type condition flag so that the download type is not hard-coded in the system process that creates the NDS.

We have populated the Notification Download Type column with a few classic examples:

Notification Download Type	Download Type Condition Flag
Inform all parties that customer is stopping service	CUSTSTOP
Send consumption history	
Check if it's OK to switch customer from current supplier	
Reject request	

You should also consider the priority of each NDS type with respect to the other types.

Once you know the different types of download notifications, you have to determine the physical method used to route the notification to the third party. Refer to *Designing Notification Download Profiles* for how to do this.

If the outbound notification may be routed via XAI, you must define an *XAI inbound service*. This is used by the XAI MPL to retrieve information about the record related to the outbound message.

Fastpath: Refer to *Configuring the System for XAI Messages* for more information on designing your XAI outgoing messages.

Designing Context Values for the NDS Type

A notification download staging record must reference some "relevant data" that allows the extract process or the XAI MPL process to retrieve the data needed to send to the external system. For example, if the person's name has changed, the NDS must reference the Person Id. When formatting the data to send to an external system, the appropriate process would take the person ID and retrieve the data needed to send to the external system.

The NDS may use either the context collection or the characteristics collection to enter this "relevant data". If you choose to use context, you may decide to define the context types for the NDS type (although it is not required).

If NDS records of this type are interfaced through XAI, then the context types and corresponding XPATH values are required for every piece of data that is required to call an application service to build the extract. The XPATH indicates the relative path in the XML document where the field value will be placed when building the XML document.

Fastpath: Refer to *Configuring the System for XAI Messages* for more information on designing your XAI outbound messages.

Configuring the System for XAI Messages

Designing Your Real Time Messages

Consider the following integration scenario. A page in your product is configured to send information to an external system real time. If the connection is down, the user is warned and may authorize that the message is sent in near real time. The following sections describe how to design the control tables needed for both scenarios.

Designing the Real Time Message XSL

During implementation, you will determine the information required to send to the external system. The real time message engine is passed all the data available in the page model as an XML document. The request XSL must be designed to map this data into a format expected by the target.

In addition, you should create two response XSLs to transform the message received from the target system.

- One XSL is used for real time responses. The XSL transforms the message into a format understood by the page service user exit. Note that any updates to a system business object as a result of the response is the responsibility of the user exit that called the real time message engine.
- One XSL is used for non-real time responses. In this case, if the response should update a system object, the response must trigger an XUS to update the system object. Therefore, the XSL must map the response into an XML request that reference the appropriate service to update the system object.

Designing the Real Time Message Route Types

For this scenario, you must create two route types.

- Create an *XAI Route Type* for the real time message. Indicate the sender, the request XSL and the response XSL used for real time responses.

Note: The sender referenced on the route type should be one that supports synchronous communication (such as an HTTP sender).

- Create an XAI route type for the near real time message. Indicate the sender, the request XSL and the response XSL used for near real time responses. Mark the route type to check the flags Receive Acknowledge and Post Response. These flags indicate to XAI that a response should be received and that additional processing is required.

Designing the Real Time Message NDS Types and Download Profile

For this scenario you must create two NDS types

- One NDS type is used for the real time message. Indicate an appropriate notification download condition. This allows the mechanism that creates the NDS to refer to this condition rather than hard-coding the NDS type.
- One NDS type is used for the near real time message. This NDS type must reference a special XAI inbound service used to indicate to the download staging sender that the XDS does not need to be created.

Note: Download Condition. The real time message engine receives the NDS type as input. It is the user exit's responsibility to determine the appropriate NDS type based on a notification download condition.

You'll need to create an entry in the service provider's *notification download profile* for each NDS type. Set the processing method to `XAI` and indicate the appropriate route type (real time or near real time) created above.

Designing the Near Real Time Response

To process a response to the near real time message, you must also design the XAI inbound service to be used to process the response.

- The XSL transformation scripts that are referenced by the inbound service should populate the appropriate XML elements to allow XAI to recognize that the inbound message is a response.
- Otherwise, the inbound service should be designed to process this message as appropriate. For example, if the response should update the status of the record that initiated the outgoing request, the inbound service should reference the appropriate schema to update the appropriate record.

Designing Near Real Time NDS Messages

As described above, near real time NDS messages require you to define:

- A download staging receiver to process records on the NDS table.
- A download staging sender to process "responses" to the download staging receiver.

The following sections identify other control table design issues.

Consider the following integration scenario. A change to a certain type of record in your product should trigger a message to an external system to inform that system of the change. This will be a near real time message and we expect an asynchronous response.

Designing the XML Request and Inbound Service

You should work with the external system to determine the information they require for this message. If a system service already exists to extract all the required information, you may use that service. However, it's possible that you would need to create a new service to extract the information you need.

Your implementers must create an appropriate service and then use the schema editor to create request and response schemas based on this service.

Note: Message ID. If you expect an asynchronous response to this message, the request and response schemas must include a private attribute to hold the `MessageID`.

Create an XAI inbound service for extracting the appropriate information and building the XML request.

Designing the Near Real Time NDS Type

Define an appropriate NDS type. The information defined here is required to successfully build the XML request.

- Indicate the XAI Inbound Service created above.

- Indicate an appropriate notification download condition. This allows the mechanism that creates the NDS to refer to this condition rather than hard-coding the NDS type.
- Use the notification download type context to identify the data required to build the request to invoke the XAI Inbound Service for this download type. In our case, context types must be defined for the business object id and the message id. Use the *XPATH* to indicate the relative page for this element in the request schema.

Context Type	XPATH
Maintenance Object ID	//ExtractInfoService/ExtractInfoHeader@MaintenanceObjectID
Message ID	//ExtractInfoService/ExtractInfoHeader/MessageID

Designing the Near Real Time Route Type

You must create an *XAI Route Type*. The route type identifies the sender for routing the message and the XSL transformation script required to transform the request into a format expected by the sender.

If the sender supports synchronous communication, you must check the Receive Acknowledge and Post Response flags to successfully process the response. Our scenario indicated that we expect an asynchronous response so we'll leave these flags unchecked.

Note: Routing Optional. If the message does not require routing, you are not required to define an XAI route type.

Designing the Response

If you expect a response to the message sent, you must also design the XAI inbound service to be used to process the response.

- The XSL transformation scripts that are referenced by the inbound service should populate the appropriate XML elements to allow XAI to recognize that the inbound message is a response.
- Otherwise, the inbound service should be designed to process this message as appropriate. For example, if the response should update the status of the record that initiated the outgoing request, the inbound service should reference the appropriate schema to update the appropriate record.

Designing Notif. Download Profiles

A notification download profile controls how the system routes notifications to third parties. You associate a notification download profile with one or more service providers. Whenever a notification is sent to a service provider, the system uses the notification download profile to determine the interface method and the format of notifications.

Note: A notification download profile corresponds with a protocol used to route notifications to service providers. If you electronically route all notifications using the same protocol (for example, all service providers in a given jurisdiction may conform to the same record formatting and routing method), you will have a single notification download profile. If you have to route notifications using different protocols to different providers, you will need one notification download profile per protocol.

Using the notification download profile, you can define which outgoing messages are sent as an XML document via the XAI tool and which are sent in batch format. When a message is designated as being sent via the XAI tool, then you typically define XAI routing information. If a message is designated as being sent via batch, then you must indicate the formatting algorithm used by the extract program.

Note: XAI Routing Information Optional. You may indicate that the message should be routed via XAI, but enter no XAI Routing information. You may do this when you are interfacing with one of your own external systems and where the system may be accessed directly. In this scenario, you are using the XAI functionality to communicate between systems using XML documents, but you don't need the routing logic within XAI. Refer to *An NDS Message That Doesn't Require Routing* for more information.

The easiest way to design a notification download profile is to choose a representative service provider and design a notification download profile for it by filling in the following matrix. After you've designed a profile, determine how

many other service providers can use it. Then design the next service provider's profile and determine who can reuse it. Repeat this process until all your service providers have a profile. Once the profiles are designed, you're ready to set up the control tables.

Background Process used for batch messages	Notification Download Type	Processing Method	XAI Route Type	Format Method
File Transfer	Inform all parties that customer is stopping service	Batch		Format Stop
	Send consumption history	Batch		Format Consumption
	Check if it's OK to switch customer from current supplier	XAI	Confirm Switch	
	Reject request	XAI	Reject	

Notice that you define the background process at the profile level, but you indicate for each download type whether it will be processed via batch or via XAI. If at least one of the processing methods is batch, then a background process must be entered, otherwise it's not applicable.

Also note that it is possible to link more than one XAI route type to each notification download type formatting method. However, this is not recommended if you expect an *asynchronous response to the NDS message*.

After you've designed notification download profiles to cover every protocol, you are ready to set up the control tables.

Fastpath: Refer to *Notification Download Background Processes*.

Setting Up Notification and Workflow Procedures

In the previous sections, *Designing Notification Upload & Workflow Procedures* and *Designing Notification Downloads*, we presented a case study that illustrated a mythical organization's workflow procedures. In this section, we'll explain how to set up the control tables to implement these procedures.

Setting Up Workflow Event Types

Workflow event types control what is done by a given workflow event. Open **Admin Menu, Workflow Event Type** to define your workflow event types.

Fastpath: Refer to *Designing Workflow Event Types* f **or more information**.

Description of Page

Enter a unique **Workflow Event Type** and **Description** for the event type.

Turn on **Allow Manual Completion** if an operator is allowed to change the status of workflow events of this type to `Complete`.

The **Event Activation Algorithm** is used by the system when it activates a workflow event of this type. Refer to *Executing Workflow Events On Their Trigger Date* and *Designing Workflow Event Types* for more information. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).

- On this algorithm, reference an Algorithm Type that is associated with workflow event activation. The system comes supplied with several sample algorithm types that should be used as a sample if you have to write a new algorithm type. Click *here* to see the algorithm types available for this system event.

The **Event Failure Algorithm** is used by the system when a workflow event of this type fails. This algorithm need only be defined if events of this type can fail. Refer to *Some Workflow Events May Fail* and *Designing Workflow Event Types* for more information. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that is associated with workflow event failure. The system comes supplied with several sample algorithm types that should be used as a sample if you have to write a new algorithm type. Click *here* to see the algorithm types available for this system event.

The **Wait Process** is the background process responsible for monitoring workflow events of this type that are in the `Waiting` state. Refer to *Waiting Events And Their Waiting Process* for more information.

Where Used

Follow this link to view the tables that reference *CI_WF_EVT_TYPE* in the data dictionary schema viewer.

Setting Up Workflow Process Templates

A workflow process template defines the workflow events that will be created when a workflow process is created using a template.

Fastpath: Refer to *Designing Workflow Process Templates* f **or more information**.

Workflow Process Template - Main

Open **Admin Menu, Workflow Process Template** to define your workflow process templates.

Description of Page

Enter a unique **Workflow Process Template** code and **Description** for the workflow process template.

The system needs to know the type of workflow process to create when a notification is uploaded that does not contain a valid External System or Notification Upload Type (these two fields are the ones that control the type of workflow process that's created to process the uploaded notification). If you create workflow process templates and label them with a **Notification Condition** of `Unknown Notification ID` or `Unknown Notification Type`, the notification upload process will create a respective workflow process when either of the above conditions are discovered. Note: most financial services create an outgoing notification rejecting the uploaded notification when such conditions transpire. The notification condition field is also available for use by your plug-in algorithms for any purpose where you need to identify a specific workflow process.

Note: The values for this field are customizable using the Lookup table. This field name is WF_PROC_COND_FLG.

Use **Comments** to describe the workflow process template.

When a workflow process is created, the system links one or more workflow events to it. The information in the **Workflow Responses** scroll defines these events and when they will be triggered. The following fields are required for each event:

Event Sequence Sequence controls the order in which the workflow events are executed. The sequence number is system-assigned and cannot be changed. If you have to insert a workflow event between two existing events, you'll have to remove the latter events, insert the new event, and then re-specify the removed events.

Workflow Event Type Specify the type of workflow event to be generated. The event type's description is displayed adjacent.

Dependent on Other Events Turn this indicator on if the trigger date of the event can only be determined after earlier events are complete. Refer to *Workflow Event Dependencies & Trigger Date* for more information. If this switch is on, you must define the events on which this response depends in the **Dependent on Other Events** grid.

Days After Previous Response Specify the number of calendar days after *the completion of the dependent events* on which the workflow event will be triggered. If this event is not dependent on the completion of other events, this field contains the number of calendar days after the *creation of the workflow process* that the related workflow event will be triggered.

When the **Dependent on Other Events** switch is on, a grid appears in which you specify the events on which this event is dependent. The following fields are required for each event:

Sequence Sequence is system-assigned and cannot be specified or changed.

Dependent on Sequence Specify the sequence number of the workflow event type on which the above workflow event depends.

Workflow Event Type The system displays the ID of dependent workflow event in this column.

Fastpath: For more information about workflow event templates, see *Setting Up Workflow Event Types*. For more information about trigger dates, see *Workflow Event Dependencies & Trigger Date*.

Where Used

A Workflow Process Profile references one or more workflow process templates. Refer to *Setting Up Workflow Process Profiles* for more information.

A Workflow Process references a workflow process template. Refer to *Workflow Process - Main* for more information.

Workflow Process Template - Template Tree

Open **Admin Menu, Workflow Process Template** and navigate to the **Template Tree** tab to view information about your workflow process template.

Description of Page

This page is dedicated to a *tree* that shows the events linked to the workflow process and information about the event dependencies. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

Setting Up Notification Upload Types

Every notification upload staging record has a notification upload type. This code is one of several fields that control the type of workflow process used to process the incoming notification. Open **Admin Menu, Notification Upload Type** to define your notification upload types.

Fastpath: Refer to *Designing Notification Upload Types* f **or more information**.

Description of Page

Enter a recognizable **Notification Upload Type** and **Description**.

Enter a value for the **Upload Condition Flag** when a system condition should trigger the creation of a notification record. Refer to *System Conditions May Trigger Notification and Workflow* for more information.

Note: The values for this field are customizable using the Lookup table. This field name is NT_UP_TY_COND_FLG.

If NUS records of this type will be associated with an **Extension** record, indicate the extension here along with its **Navigation Option** to allow a user to navigate to the correct extension record when viewing the notification upload staging record. Refer to *Navigating to Related NUS Extension* for more information.

Where Used

A *Notification Upload Staging* record must reference a notification upload type.

A *Workflow Process Profile* references one or more notification upload types.

Setting Up External Systems

Every notification upload staging record references the system that sent the message. The external system is one of several fields that control the type of workflow process used to process the incoming notification. Refer to *External Systems* for more information.

Setting Up Workflow Process Profiles

The system uses a workflow process profile to determine the type of workflow process to create for incoming notifications sent by the service provider. A workflow process profile is associated with one or more service providers. Open **Admin Menu, Workflow Process Profile** to define your workflow process profiles.

Fastpath: Refer to *Designing Workflow Process Profiles* **f or more information**.

Description of Page

Define a unique ID and **Description** for each workflow **Process Profile**.

The information in the grid defines the type of workflow process that will be created for each **Notification Upload Type**. The type of workflow process may differ depending on special criteria. For example, you may have a different workflow process if the customer is industrial. You must define `Default` criteria in case none of the override criteria are met (the `Default` criteria should have the lowest priority). You MAY have override criteria if different situations result in different types of workflow processes.

The following fields are required for each criterion:

Priority The priority controls the order in which the system determines if the respective workflow process should be used to process notifications of a given type. Higher priorities are checked before lower priorities.

Note: The values for this field are customizable using the Lookup table.

Criteria Algorithm Select the algorithm to be used to check if the workflow process should be initiated for notifications of a given type. If a condition is met, a workflow process is created using the associated workflow process template.

If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that determines if an incoming notification should be processed using the associated **Workflow Process Template**. The system comes supplied with a sample algorithm type called that should be used as a sample if you have to write a new algorithm type. Click *here* to see the algorithm types available for this system event.

Note: Important! You must have at least one entry in this collection otherwise the system will not start a workflow process when an incoming notification of this type is received. This entry should have the lowest priority code and should reference a **Criteria Algorithm** that references the default workflow criteria algorithm type.

Workflow Process Template Specify the workflow process template to use to process incoming notifications identified with the notification upload type.

Fastpath: Refer to *Designing Workflow Process Criteria* for more information.

Setting Up Notif. Download Types

Every notification download staging record has a notification download type. This code controls the format of the record that is sent to a service provider. Open **Admin Menu, Notification Download Type** to define your notification download types.

Fastpath: Refer to *Designing Notification Download Types* f **or more information**.

Notification Download Type - Main

Every notification download staging record has a notification download type. This code controls the format of the record that is sent to a service provider. Open **Admin Menu, Notification Download Type** to define your notification download types.

Description of Page

Enter a unique **Notification Download Type** and **Description.**

Specify the **XAI In Service Name** that will be called for this NDS type if the download profile formatting method indicates that this download type will be processed by XAI.

Enter a value for the **Download Type Condition Flag** when a system condition should trigger the creation of a notification record. Refer to *System Conditions May Trigger Notification and Workflow* for more information.

Note: The values for this field are customizable using the Lookup table. This field name is NT_DWN_TY_COND_FLG.

Indicate the relative **Priority** for processing NDS records of this type with respect to other types. This value defaults to `Priority 90 - Lowest`.

Note: The values for this field are customizable using the Lookup table. This field name is NT_DWN_TY_PRIO_FLG.

Notification Download Type - Context

If the notification download staging record is communicated to the external system through XAI, the NDS context is used to help build the XML document.

Description of Page

The context collection functionality allows you to define a collection of **Context Type** sthat should be defined for a notification download staging record of this type. When the NDS record is created, with its collection of these Context Types, the Context Values would correspond to system data related to this NDS record. In addition, you may specify a **Context Value** if this is a constant value for NDS records of this type.

The **XPATH** is used by XAI NDS Types. For each context type, use the XPATH to indicate the relative path in the XML document where the field value will be placed when building the XML document.

Where Used

A Notification Download Staging record must reference a notification download type. Refer to *Process X - Populate Notification Upload Staging* for more information.

A Notification Download Profile references one or more notification download types. Refer to *Setting Up Notification Download Profiles* for more information.

XAI Route Type

Use this page to define information required by the XAI tool to send outgoing messages. Navigate to this page using **Admin Menu, XAI Route Type**.

Description of Page

Enter the **XAI Route Type**, which defines the information required for outgoing messages, which use the XAI tool. Enter a **Description** for this route type.

The **XSL Request** is the schema used to transform information from the format produced by the system to a format understood by the sender, who receives a message of this type.

The **XSL Response** is the schema used to transform information from the format sent to us by the sender, who responds to this message into a format understood by the system.

Use the **XAI Sender** to indicate where messages of this type should be sent.

Fastpath: Refer to *XAI Sender* for more information.

Check the **Receive Acknowledge** box if the system expects to receive a synchronous response to outgoing messages of this type.

Check the **Post Response** box if a synchronous response to an outgoing message requires something to occur in the system. If the box is checked, a response to a message of this type causes an XAI upload staging record to be created. That record is processed along with other uploaded messages, to invoke an appropriate service.

Where Used

All outgoing messages are sent through the notification download staging record. Information related to the formatting of messages is defined on a notification download profile.

Setting Up Notif. Download Profiles

A notification download profile controls how the system routes notifications to service providers. You associate a notification download profile with one or more service providers. Whenever a notification is sent to a service provider, the system uses the notification download profile to determine the interface method and the format of notifications.

Open **Admin Menu, Notification Download Profile** to define your notification download profiles.

Fastpath: Refer to *Designing Notification Download Profiles* for more information.

Description of Page

Define a unique ID and **Description** for each **Download Profile**.

Use **NDS Extract Process** to define the background process that creates notification download records and interfaces them out of the system.

The information in the scroll controls how the **Extract Process** formats an interface record for each **Notification Download Type**. In addition to defining a **Description** and **Comments**, you must define the **Processing Method**. The valid values are `XAI` and `Batch`.

If your processing method is `Batch`, you need to define the **Notification Format Algorithm** that actually formats the interface record. If you haven't done so already, you must set up this algorithm in the system. To do this:

- Create a new algorithm (refer to *Setting Up Algorithms*).
- On this algorithm, reference an Algorithm Type that is associated with workflow event failure. The system comes supplied with several sample algorithm types that should be used as a sample if you have to write a new algorithm type. Click *here* to see the algorithm types available for this system event.

If your processing method is `XAI`, you may define one or more **XAI Route Types**. The *XAI Route Type* describes how the message should be formatted and the destination of the message.

Chapter 15

Security Addendum

Topics:

This chapter is an addendum to the general *Defining Security and User Options* chapter. This addendum describes security functionality that is specific to Oracle Revenue Management and Billing.

Implementing Account Security

Caution: Important! This section assumes you understand *The Big Picture of Row Security*.

When you create an account, you must define which users can access the account's information. For example,

- If you have customers in two geographic territories, you may need to restrict access to accounts based on the office that manages each territory. For example, only users in the northern office may manage accounts in the northern territory.

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

Fastpath: Refer to *If You Do Not Practice Account Security* for setup instructions if your organization doesn't practice account security.

Note: Account security may also affect persons. Refer to *Persons Can Also Be Secured* for how access to person information is also restricted by account security.

The topics in this section describe how to implement account security.

Persons Can Also Be Secured

It's important to be aware that persons can also be secured as a result of "account security". It works like this:

- If a person is linked to at least one account, users will not be allowed access to the person (or the person's related information) unless they have access to at least one of the person's accounts.
- If a person is not linked to any accounts (a rare situation), any user may access the person.

Note: How are persons linked to accounts? A person is linked to an account when an account is created using the methods described under *Order User Interface Flow*. In addition, you may manually link and unlink persons from account using the *Account - Person* page.

Data Becomes Invisible When Access Is Restricted

The following points summarize the impact of a user not having access to an account.

Account Security and Control Central

This section summarizes the impact of account security on the **Insurance Control Central** and **Collection Control Central** screens:

- Searches are affected as follows:
  - An account will only be visible if a user has access to the account's access group.
  - Persons that are not linked to accounts will be visible to all users.
  - If a person is linked to an account, the person will only be visible if the user has access to at least one of the person's accounts.
- The alerts that highlight the existence of "multiple relationships" are not impacted by account security. Specifically:

    - The alert `Person has multiple accounts` will appear if the selected person is linked to multiple accounts, even if the user doesn't have access to every account. Note well, the person couldn't have been selected if the user didn't have access rights to at least one account.
- Only accounts to which the user has access will be displayed in the person tree.
- Only accounts to which the user has access will be displayed in the account tree.
- All other pages contain information related to the current account context. The current account context can never reference an inaccessible account and therefore these pages are not impacted by account security.

Account Security and Searches (and Maintenance Pages)

Searches are the gateway to the information that appears on maintenance pages. In general, account-related information is suppressed when a user doesn't have access rights to the account. This suppression is true for rows that directly reference an account AND for rows that indirectly reference an account. For example:

- A user can only see bills associated with accounts to which they have access rights.
- A user can only see financial transactions associated with contracts that are, in turn, associated with accounts to which they have access rights.

Account Security and To Do Lists

Account security does NOT impact the information that appears in a user's ToDo list. Rather, we have assumed that your ToDo roles (and the users assigned to these roles) are consistent with your account security requirements. This can result in anomalies. For example, it's possible for a supervisor to assign a bill segment error to a user who doesn't have access to the bill segment's account. This user will then see the related ToDo entry in their Bill Segments In Error ToDo list. However, when they drill down on the entry, account security will manifest itself (i.e., the user won't be able to display the bill segment that's in error). This happens because the drill down causes the bill segment search logic to execute. This logic inhibits the selection of bill segments if the user can't access the related account.

To minimize these anomalies, we recommend the following:

- Setup *To Do Roles* consistent with your Data Access Roles.
- Setup *Account Management Groups* that are consistent with your Access Groups.
- Setup default To Do Roles on your Account Management Groups for each *ToDo type*.

Restricted Transactions

The following table lists all transactions that have some type of account security. The following notation is used to describe the type of account security:

- **Account-oriented.** This notation is used if the respective transaction uses basic account security (i.e., the user must belong to at least one data access role that has access to the account's access group in order to see the information).
- **Person-oriented.** This notation is used if the respective transaction uses person-oriented account security. Refer to *Persons Can Also Be Secured* for more information.
- None of the above. Some unusual transactions have unusual implementations of account security. These are described below.

Transaction	Type of Account Security
Account	Account-oriented
Account Bill / Payment History	Account-oriented
Account Financial History	Account-oriented

Transaction	Type of Account Security
Account Interval Info	Account-oriented
Account Payment History	Account-oriented
Account Person Replicator	Account-oriented
Account Contracts for Debt Class	Account-oriented
Adjustment	Account-oriented
Adjustment Calculation Line Characteristics	Account-oriented
Bill	Account-oriented
Bill Print Group	Person-oriented
Bill Segment	Account-oriented
Billable Charge	Account-oriented
Budget Review	Account-oriented
Case	Account-oriented, Person-oriented
Collection Agency Referral	Account-oriented
Collection Process	Account-oriented
Contract Option	Account-oriented
Control Central	Account-oriented, Person-oriented
Customer Contact	Person-oriented
Declaration	Account-oriented
Deposit Review	Account-oriented
Financial Transaction	Account-oriented
Financial Transaction on a Bill	Account-oriented
Financial Transaction on a Payment	Account-oriented
Interval Profile	Account-oriented (for contract-specific profiles)
Loan	Account-oriented
Match Event	Account-oriented
Multi-Cancel/Rebill	Account-oriented
Non-billed Budget	Account-oriented

Transaction	**Type of Account Security**
Order	Account-oriented, Person-oriented
Overdue Process	Account-oriented
Payment	Account-oriented
Payment Arrangement	Account-oriented
Payment Arrangement for Bills	Account-oriented
Payment Event	The user must have access to ALL accounts linked to the payment event.
Payment Event QuickAdd	The user must have access to ALL accounts linked to the payment event(s).
Payment QuickAdd	Account-oriented
Payment / Tender Search	Account-oriented
Person	Person-oriented
Quote	Account-oriented
Contract Billing History	Account-oriented
Contract Cash Accounting Balance	Account-oriented
Contract Financial History	Account-oriented
Contract	Account-oriented
Service Credit Event	The user must have access to ALL accounts linked to the service credit membership that has the service credit event.
Service Credit Membership	The user must have access to ALL accounts linked to the service credit membership.
Start/Stop	Account-oriented
Statement	The user can see the statement if they have access to at least one account on the statement's statement construct.
Statement Construct	The user can see the statement if they have access to at least one account on the statement construct.
Terms of Service	Account-oriented (User must have access to at least one account linked to the contract collection in the TOS.)
Write Off	Account-oriented
Write Off Process	Account-oriented

Account Security Case Studies

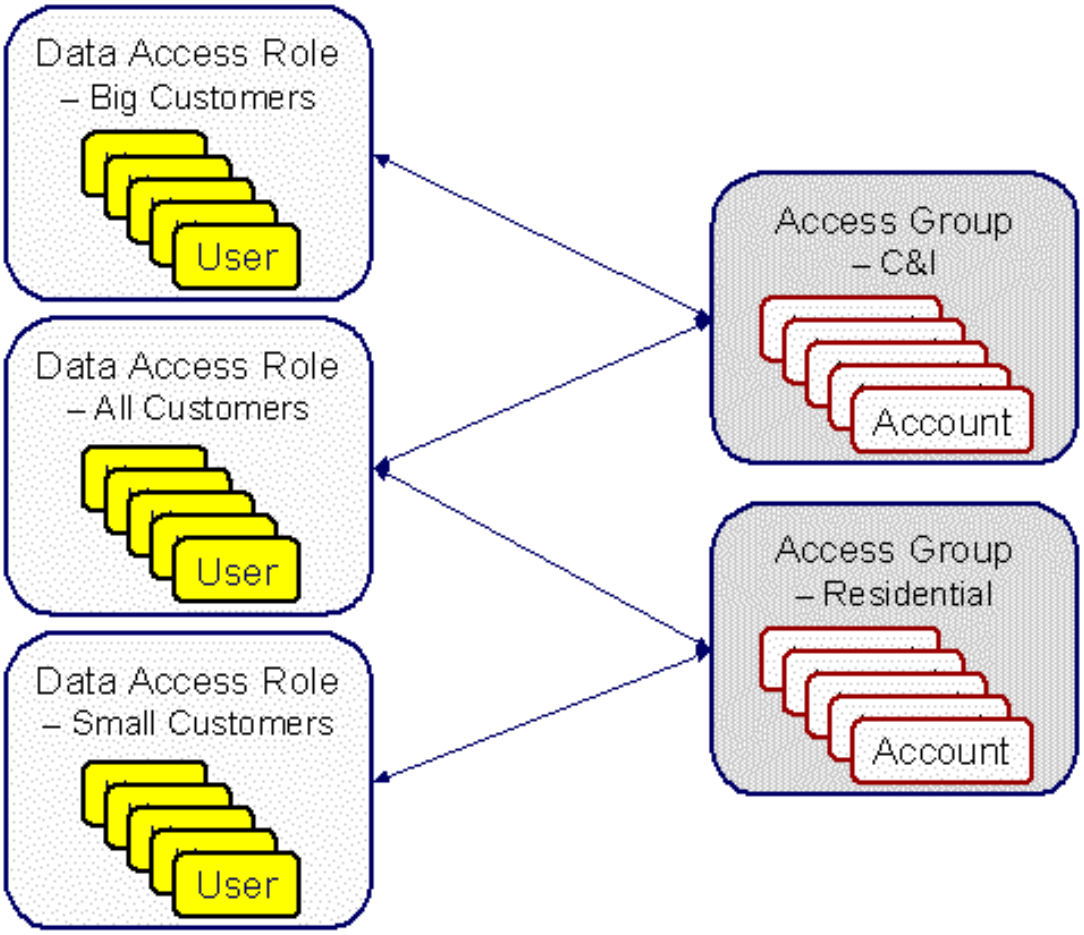
The topics in this section contain examples of how to implement account security. Use these examples to form an intuitive understanding of these objects. Once this intuition is obtained, you'll be ready to design the account security objects for your own company.

Securing Accounts Based On Customer Class

Assume the following security requirement exists:

- You have two broad groups of accounts:
  - Financial services accounts.
  - Other accounts.
- Users can be classified as have one of the following access rights:
  - May access all accounts.
  - May only access financial services accounts.
  - May only access other accounts.

The following diagram illustrates the access groups and data access roles required to implement these requirements:



Notice the following about the above:

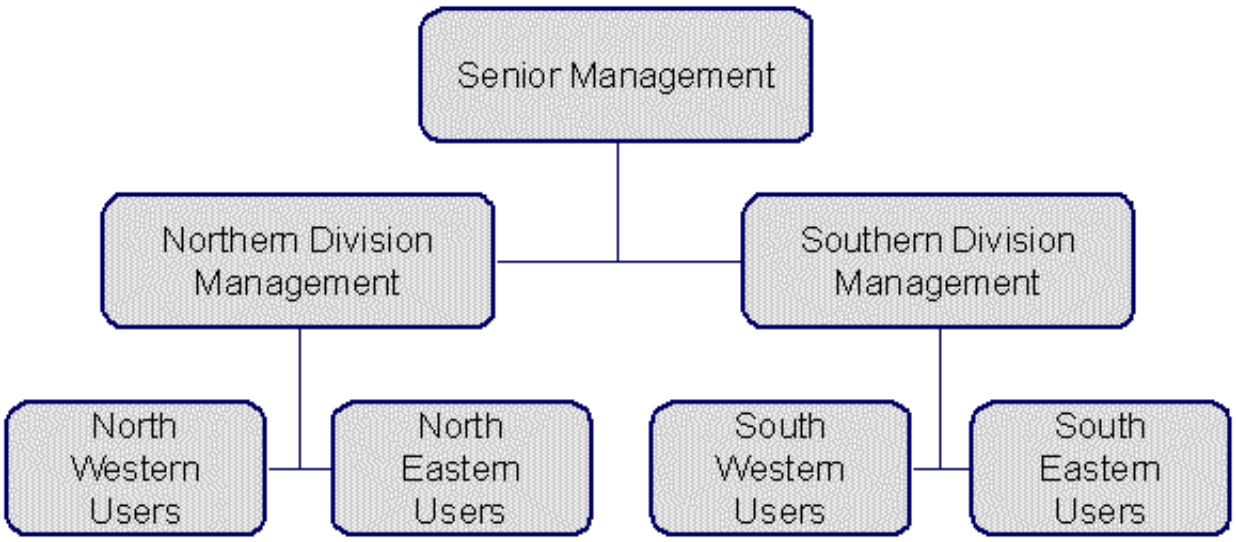
- There are 2 access groups because access to accounts is based on whether the account is considered to be residential or commercial/industrial.
- The Big Customers data access role is only linked to the C&I access group.
- The Small Customers data access role is only linked to the Residential access group.
- The All Customers access role is linked to both the C&I and Residential access groups. Users with this role can therefore access all accounts.

Securing Accounts Based On Region

Assume that accounts are classified as belonging to one of the following regions:

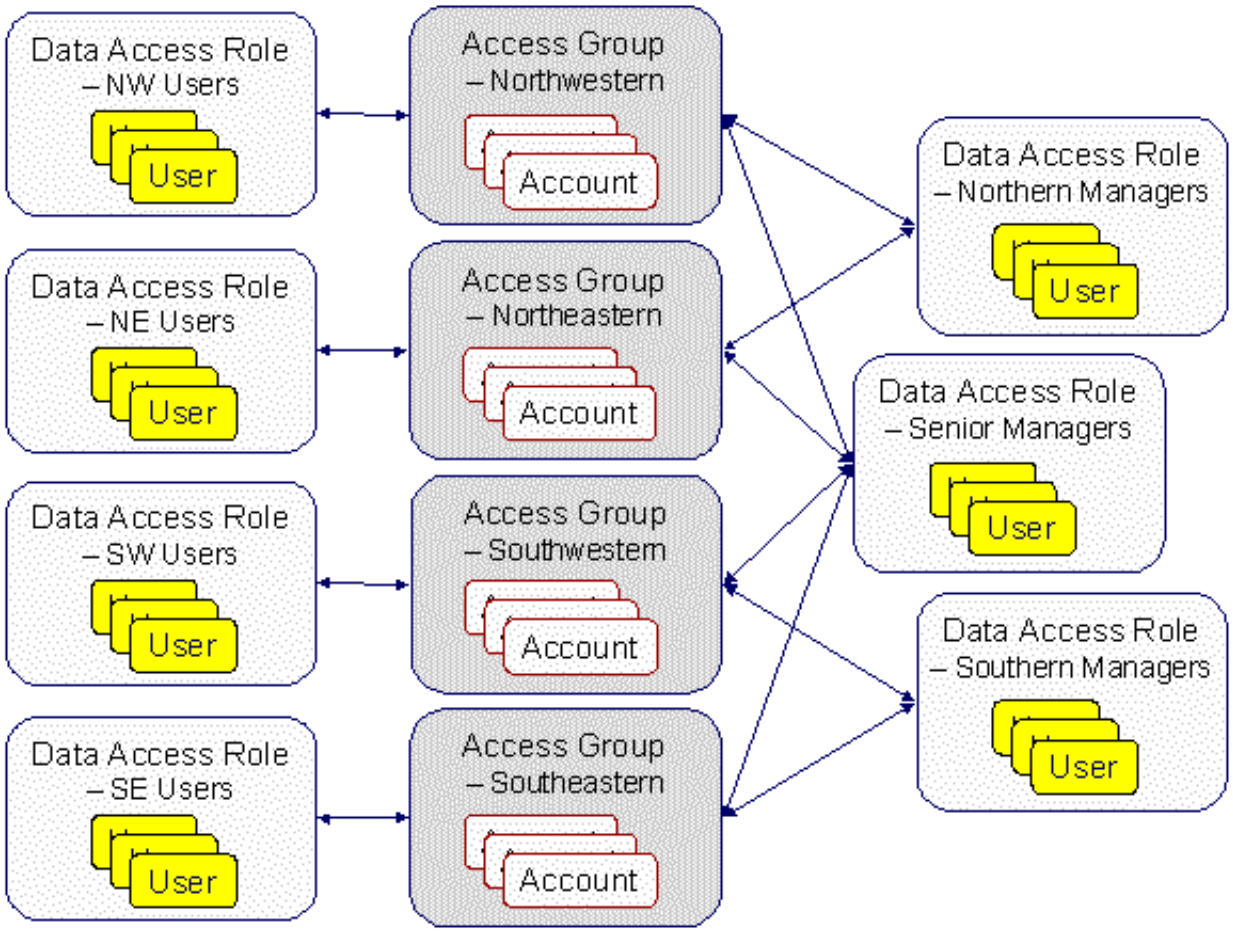
- Northwestern
- Northeastern
- Southwestern
- Southeastern

Assume the following company hierarchy exists:



- Senior Management has access to all customers
- Northern Division Management has access to all customers in the Northwestern and Northeastern divisions.
- Southern Division Management has access to all customers in the Southwestern and Southeastern divisions.
- Northwestern Users have access to all customers in the Northwestern division.
- Northeastern Users have access to all customers in the Northeastern division.
- Southwestern Users have access to all customers in the Southwestern division.
- Southeastern Users have access to all customers in the Southeastern division.

The following diagram illustrates the access groups and data access roles required to implement these requirements:



Notice the following about the above:

- There are 4 access groups because access to accounts is based on the region in which they are located (and there are 4 regions).
- There are 7 data access roles because each component of every layer of the access hierarchy requires a separate data access role.

The Default Access Group

There are two ways an access group can be assigned to an account:

- The base package will default an account's access group based on the user who adds the account. It uses the user's *default access group* to do this.
- If you can conceive of a rule to assign an appropriate access group to a newly added account, you can have your programming staff introduce a user exit to the account row program to implement this rule. For example, a user exit could be developed that assigns an access group to an account based on its customer class. The name of the user-exit is AFTER-MOVE-CORR-ADD and the name of the account row program is CIPCACCR.

Caution: Regardless of the method used to assign an account's access group, please be aware that the user who adds an account must have access to this access group.

Note: Subsequent changes to an account's access group. A user may change an account's access group to any access group to which they have access.

If You Do Not Practice Account Security

If you do not restrict access to accounts (i.e., all users can access all accounts), you must set up one access group and one data access role and then indicate all users are part of this role. You should also define the access group as the default access group on all of your users (so that new accounts are all labeled with this access group).

Masking Sensitive Data

Refer to *Encryption and Masking* for instructions on how to configure the system to mask sensitive data like a customer's social security number or bank account number.

Chapter 16

Defining Overdue Processing Options

Topics:

The system periodically monitors how much your customers owe to ensure they haven't violated your collection rules. When a violation is detected, the system initiates the appropriate activities (e.g., letters, disconnect notices, collection agency referrals, and eventually write off). The topics in this section describe how to configure the system to manage your overdue processing requirements.

Note: Collecting on unpaid bills. The overdue processing module has been designed to collect on virtually anything from an unpaid bill to an unmatched financial transaction. You tell the system what you collect on by configuring the various overdue processing control tables. In this release, the base-package is delivered with algorithms that support collecting on overdue bills. If your organization practices balance-forward accounting (i.e., collection is based on overdue contract balances), you will not use this functionality. Rather, you will use the functionality described under *Defining Credit and Collections Options*.

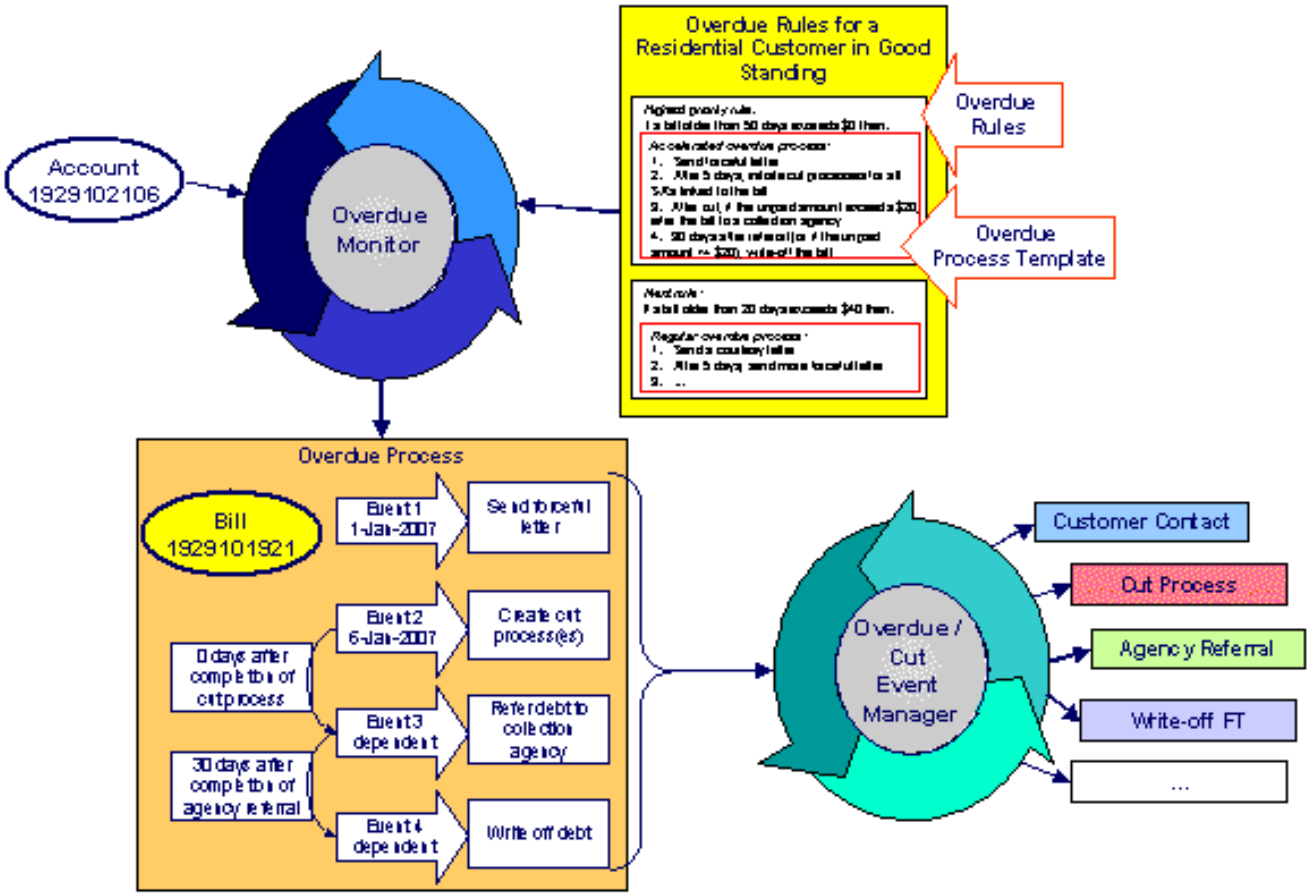
Caution: Straightforward rules = straightforward set up. Setting up this module is as challenging as your organization's collection rules. If you have simple rules, the set up process will be straightforward. If your rules are complicated (e.g., they differ based on the type of customer, the age of debt, the type of service, etc.), your setup process will be more challenging.

Case Study - Collecting On Overdue Bills

The following topics introduce a case study that describes how overdue processing can be used to collect on overdue bills. This is just an example as virtually every aspect of overdue processing is configurable. Use this case study to familiarize yourself with the overdue processing core concepts.

Monitoring Overdue Bills

The following diagram illustrates the objects and processes involved with collecting overdue bills.



There are many important concepts illustrated above:

The Overdue Monitor checks if your accounts have bills that violate your overdue rules	The *Overdue Monitor* is a background process that periodically reviews your account's financial obligations. Note well: every account belongs to a collection class. There are two types of collection classes: those whose accounts are monitored by the *Account Debt Monitor*, and those that are monitored by the Overdue Monitor. This chapter describes the Overdue Monitor.

Overdue rules define when and how unpaid bills are collected	An account's *collection class overdue rules* have algorithms that monitor an account's financial obligations. These algorithms are invoked by the Overdue Monitor when it's *time to review* an account's obligations. These algorithms can contain any type of criteria. However, most are defined using a combination of a threshold age and monetary amount. For example, a classic algorithm would check if a bill has unpaid financial transactions more than 20 days old that exceed $50. In the case of bill-oriented collection, the monitoring algorithms look at each of the account's bills to determine if they are paid. Note, a bill is considered paid if its financial transactions (FTs) are linked to a `balanced` match event. If a monitoring algorithm finds an unpaid bill, it can check if it old enough (and large enough) to be considered a violation. When you set up a monitoring algorithm, you define the type of overdue process that should be created when an overdue bill is detected. You do this by defining the appropriate "overdue process template".
An overdue process template defines how to handle an overdue bill	An *overdue process template* contains one or more *overdue event types*. These define the number and type of events that are created to prod the customer to pay. For example, you might set up an overdue process template with event types to send a series of letters followed up by a call. The overdue process template has contains the rules defining *when events are activated*. The specific action that's performed by an overdue event is controlled by the `Activation` algorithm defined on its event type. Refer to *Overdue Event Type - Main* for a list of the various `Activation` algorithms delivered with the base package.
Multiple objects can be associated with a single process	The above diagram shows a single bill linked to an overdue process. It should be noted that an overdue process is capable of referencing multiple bills (or other objects). Note well: while a single overdue process can reference many overdue objects, all such objects must be of the same type. For example, you cannot commingle bills and contracts under a single overdue process. The type of object managed by an overdue process is defined on its *overdue process template*.
If a customer pays the bill, the overdue process is cancelled	If an overdue bill is paid, the *overdue process is canceled* real-time. You control if and how an overdue process is cancelled by setting up the appropriate rules on the *overdue process template*.
The Overdue / Event Manager activates and triggers overdue events	The *Overdue Event Manager* is a background process that activates overdue events on the appropriate date. On this date, the event's `Activation` algorithm(s) are called. This Overdue Event Manager also has the responsibility of recursively activating later events that are dependent on the completion of earlier events.

Events can be activated real-time	*Overdue Process - Main* has a button that allows users to activate (and recursively trigger) overdue events online / real-time. This means you don't have to wait for a batch job to activate events.
Overdue events can wait for related activities to complete	As described above, an overdue event's `Activation` algorithm can create virtually any object. What wasn't explained is that the event can be set up to *wait* for the ancillary object to finish before it completes. For example, an event can create a To Do entry and wait for it to complete before the next event is triggered. You can introduce plug-ins to create and wait on virtually any object. While an overdue event is in the `Wait` state, the Overdue Event Manager monitors the state of the related object(s). When the related object completes, the event is transitioned to the Complete sate (thus triggering dependent overdue events). Please see *Some Events Wait For Something Before Completing* for more information.

How Does The Overdue Monitor Work?

This section describes how the Overdue Monitor background process (batch control: `C1-ADMOV`) uses your overdue rules to collect overdue debt.

Note: Recommendation. We recommend that you familiarize yourself with the concepts described in the *case studies* before reading this section.

Note: Collecting overdue bills. The examples in the following section frequently refer to how the Overdue Monitor is configured for an organization that collects on unpaid bills. It should be noted that these are just examples as the Overdue Monitor can be used to collect on virtually anything (if you create the appropriate plug-in algorithms).

Different Overdue Rules For Different Customers

The Overdue Monitor uses rules to control how it monitors an account's debt. The system allows you to define different rules for different combinations of collection class, division and currency code. For example,

- You probably have different collection rules for different jurisdictions (i.e., divisions). For example, if you have customers in different states / provinces, you may have different rules applied to each jurisdiction's debt. The **division on each customer's account** defines the jurisdictional rules applied to the account's debt.
- You probably have different collection rules for different customer segments. For example, bills for large customers might be overdue if they are more than 10 days late, whereas small customers might have 24 days. You differentiate your customers in respect of the overdue via the **collection class on the customers' accounts**. An account's initial collection class is defaulted from its customer class. You may override an account's collection class at will.
- You will have different criteria for every currency in which you work because the monitoring process always compares a customer's debt against some value and this value must be denominated in the customer's currency. A customer's currency is defined using a **currency code on the account**.

The above means that every combination of division, collection class, and currency code can have different rules. The following matrix is used to illustrate a sample organization's rules (note, we assume a single currency and therefore avoid the third dimension):

Account's Collection Class **Account's Division**	**Commercial Customer**	**Residential Customer**
North	Highest Priority: If a bill exists with unpaid FT's > $0 that is older than 45 days, create the commercial 45 days late overdue process. Next Priority: If a bill exists with unpaid FT's > $100 that is older than 30 days, create the commercial 30 days late overdue process.	Highest Priority: If a bill exists with unpaid FT's > $0 that is older than 50 days, create the accelerated overdue process for residential customers. Lower Priority: If a bill exists with unpaid FT's > $25 that is older than 25 days, create the courtesy reminder overdue process for residential customers.
South	...	...

Notice that there can be multiple criteria for each cell in the matrix. What differentiates one criteria from another is its priority. The higher priority criteria will be compared first. If the debt violates the criteria, the overdue process is initiated and no further comparisons are performed.

Overdue Rules Are Embodied In Algorithms

Your organization's overdue rules are defined in algorithms plugged in on *Collection Class Overdue Rules* (in the `Overdue Monitor Rule` system event). When the Overdue Monitor analyzes an account, it uses the rules associated with the account's collection class, division and currency code. To analyze an account, it simply invokes these algorithms in sequence order, i.e., the lower the sequence, the higher its priority.

An `Overdue Monitor Rule` algorithm has two responsibilities:

- it determines if an account violates its overdue rules,
- if so, it creates one or more overdue processes using an *overdue process template*

When Is An Account Monitored?

The Overdue Monitor determines if an account violates your overdue rules at least every X days, where X is defined on the *Customer Class - Controls* associated with the account's customer class and division (in the field Min Credit Review Freq (Days)).

In addition, the Overdue Monitor examines an account's debt as follows:

- X days after an account's bill due date (X is defined in the field Credit Review Grace Days on *customer class control*).
- If a payment is canceled with a cancellation reason that indicates non-sufficient funds.
- If a payment arrangement is broken (assuming you use the base package's break payment arrangement plug-in).
- If a promise to pay is broken. Refer to *The Promise To Pay Monitor* for more information.
- If a *match event* is added, changed or deleted.
- When a written off bill is *reversed*.

Note: Additional events. Your implementation can have other events trigger the analysis of an account by the Overdue Monitor. To do this, add logic to insert a row on the CI_ADM_RVW_SCH table when the event occurs. This row simply references the account ID to be reviewed and the desired review date.

Collection Class Defines If And How Accounts Are Monitored

As described above, every account references a collection class. The collection class defines if and how its accounts are monitored. There are three options:

- The accounts are monitored by the *Account Debt Monitor*
- The accounts are monitored by the Overdue Monitor (this is described in this chapter).
- The accounts are not monitored for overdue debt.

Note: Migration. If you plan to migrate from the Account Debt Monitor (ADM) method of collection to the Overdue Monitor method, a special Overdue Monitor algorithm (*C1-ACT-CSW*) has been supplied that will skip accounts that are eligible for review by the Overdue Monitor if they have an active collection or write-off process. This algorithm should be plugged in the applicable Collection Class Overdue Rules so that it will be invoked first. This allows accounts with active ADM-oriented collection activities to work themselves through the system. When an account no longer has any active ADM-oriented activities, monitoring responsibilities will be assumed by the Overdue Monitor.

The Big Picture Of Overdue Processes

As described above, the Overdue Monitor subjects your accounts to overdue rules. If a rule is violated, an overdue process is created. The topics in this section provide background information about overdue processes.

How Are Overdue Processes Created?

As described *above*, the system creates an overdue process when an account violates your overdue rules. In addition, a user can manually create an overdue process at their discretion.

The Components Of An Overdue Process

The following topics describe the major components of an overdue process.

Overdue Objects

When an overdue process is created, the system links the overdue object(s) to the process. For example, if an overdue bill is detected, the bill is linked to the overdue process.

When you set up an *overdue process template*, you define the type of object it collects on by defining the *foreign key characteristic type* used to reference the object. For example, when you set up an overdue process template to collect on bills, you define a foreign key characteristic type that references the bill object.

Overdue Events

An overdue process has one or more overdue events. These events are the actions designed to encourage the customer to pay. For example, you might set up overdue events that:

- Send letters (via the creation of a customer contact)
- Create To Do entries
- Impact the account's credit rating
- Create a case (e.g., to seize the customer's assets)
- ... (the list is only limited by your imagination as algorithms are used to perform the event's actions)

You define the number and type of events by configuring *overdue process templates*. When the system creates an overdue process, it copies the events defined on the specified template.

It's important to note that all overdue events are created when the overdue process is created. A separate background process, the *Overdue Event Manager*, is responsible for activating, monitoring, and triggering overdue events.

Activation of an event causes the system to do whatever the event indicates (e.g., send a letter, send a To Do entry to a user, refer debt to a collection agency, write-off debt, etc.).

Overdue Log

Every overdue process has a log holding its history. Entries are added to the log when meaningful events occur, for example:

- When the process is created, a log entry is created to describe why the process was started.
- When an overdue event is activated, a log entry is created. These entries frequently contain a foreign key to the object that the event created so that users can easily drill down to the object from the log. For example, if an event creates a To Do entry, the To Do entry's foreign key is placed on the log and this allows a user to drill down on the log entry to see the To Do entry.
- When a process is canceled, a log entry is created to describe the circumstances of the cancellation (e.g., manual versus automated).
- Users can manually add log entries (you might want to think of these as "diary" entries) as desired.
- ...

Many of the base-package algorithms involved in overdue processing insert log entries so that a thorough audit trail is maintained. These algorithms have been designed to allow you to control the verbiage in each log entry by defining the desired message number using an algorithm parameter.

The log is viewable on the *Overdue Process - Log* page.

Note: More than just an audit trail. Please note that the log entries are more than just an audit trail. The system makes use of the log entries to know what it did. For example, when an overdue event needs to monitor the state of the To Do entries that it created, it uses the log to determine the identity of these To Do entries.

Experimenting With Alternative Overdue Process Templates

The system allows you to determine the efficacy of proposed overdue process templates using a small subset of customers before implementing the templates on the entire customer base. We use the term "champion / challenger" to reference this functionality.

We'll use an example to explain. Let's assume your prevailing overdue process template for residential customers starts with a "gentle reminder" letter followed 10 days later by a letter threatening collection agency referral if payment is not received. You may want to experiment with the impact of a change to this template. For example, you may want to change the "gentle reminder" to something more assertive and follow this up 5 days later with an even sterner warning. You can use the "champion / challenger" functionality to perform this experiment.

The following points describe how to implement "champion / challenger" functionality:

- Set up a "challenger" overdue process template for each template that you want to experiment with.
- Insert a new **Champion/Challenger** option on the `Overdue Processing` *Feature Configuration* for every champion template. Each option's value defines:
  - the "champion" overdue process template code
  - the "challenger" overdue process template code
  - the percentage of the time the system should use the "challenger" template
- Keep in mind that you can only experiment with one challenger template per champion template. For example, let's assume you have two prevailing overdue process templates - one for residential customers and another for commercial customers. You can experiment with different challenger templates for the residential and commercial templates. However, you cannot experiment with two different challenger templates for the residential champion template (i.e., a champion template can have 0 or 1 challenger template).

After setting up the above, your implementation's *Overdue Rule Plug-In* may include logic to use the challenger template X% of the time rather than the champion template. The sample plug-in provided in the base product, called *C1-CB-CR-RAT*, includes this logic.

Overdue Process Information Is Overridable

"Overdue process info" is the concatenated string of information that summarizes an overdue process throughout the user interface. The base-package logic constructs this string by concatenating the following information:

- The description of its overdue process template
- Its status
- For `active` processes, the number of days since it was created. For `inactive` processes, the number of days since it was inactivated.
- For `active` processes, the unpaid amount of the objects being collected

If you'd prefer a different info string, you can develop a new algorithm and plug-it in on your *overdue process templates*. This design allows some / all overdue process templates to have an override info string.

Original and Unpaid Amounts

There are two amounts associated with each overdue object linked to an overdue process: its Original Amount and its Unpaid Amount. These amounts are used throughout overdue processing.

You control how these amounts are calculated by defining the appropriate algorithm on your *overdue process templates*. For example, you can plug in a base-package algorithm (*C1-CUAOA*) if you collect on overdue bills and the original amount is the amount of financial transactions linked to the bill, and the unpaid amount is the sum of financial transactions that are not linked to `balanced` match events.

How Are Overdue Processes Cancelled?

A user may cancel an overdue process at their discretion, online / real-time using *Overdue Process - Main*.

The system will automatically cancel an overdue process when the object(s) associated with the overdue process are sufficiently paid. Exactly when the system checks if an overdue process should be cancelled is dependent on your organization's billing and accounting rules. For example, if you practice *open-item accounting*, you'd want to analyze an account's active overdue processes whenever a match event is added, changed or deleted (as match events are the only objects that impact if debt is considered paid in an open-item world). The base-package supports this specific example. If you need additional events to check if an overdue process should be canceled (e.g., the creation of a promise to pay), a base-package change MAY be necessary. Please check with customer support if you have questions.

Two algorithms plugged-in on the *overdue process template* handle the cancellation:

- The `Cancel Criteria` algorithm is responsible for determining if an overdue process should be canceled. Algorithms of this type analyze the outstanding debt on the objects linked to the overdue process and indicate whether a process can be cancelled.
- The `Cancel Logic` algorithm is responsible for actually canceling the process. The logic involved in cancellation can be quite sophisticated as canceling an overdue process can result in the cancellation of its pending events.

Note: Why two algorithms? The reason two algorithms are involved in cancellation is because we want the cancellation logic to be encapsulated in one place so it can be called during both manual and automated cancellation.

Note: Different logic for different templates. Because both the `Cancel Criteria` and `Cancel Logic` algorithms are plugged-in on the overdue process's template, you can have different cancellation criteria and logic for different templates.

Overdue Processes Are Created From Templates

As described above, you set up *overdue process templates* to define the types of events and when they are executed. When an overdue process is created, its events are created by copying the event types from an overdue process template. The remaining topics in this section provide background information to assist you in setting up your templates.

The Big Picture Of Overdue Events

This section describes the various types of overdue events and their lifecycle.

How Are Overdue Events Created?

Overdue events are created as follows:

- The *Overdue Monitor* invokes `Overdue Monitor Rules` to periodically check your accounts (refer to *Overdue Rules Are Embodied In Algorithms* for how this works). An `Overdue Monitor Rule` creates an overdue process when an account violates your overdue rules. The overdue process has one or more overdue event(s). The number and type of events is controlled by the overdue process template specified on the `Overdue Monitor Rule`.
- Users can create an overdue process manually on *Overdue Process - Main*. To do this, they specify an overdue process template. The number and type of overdue events is defaulted from the template.
- An overdue event may be manually added to an existing overdue process by a user on *Overdue Process - Events*.

Note: Bottom line. Most overdue events are created by the system when it creates an overdue process for delinquent obligations. If you need to create an ad hoc overdue event, you can either create an overdue process whose template contains the desired event OR link the desired event to an existing process.

Overdue Events Can Do Many Things

An overdue event can perform a wide number of activities as the logic is embodied in an algorithm. The following points describe how this works:

- Every overdue event references an *overdue event type*.
- The overdue event type, in turn, references an `Event Activation` algorithm.
- The `Event Activation` algorithm is invoked when the event is *triggered*.

Overdue Event Information Is Overridable

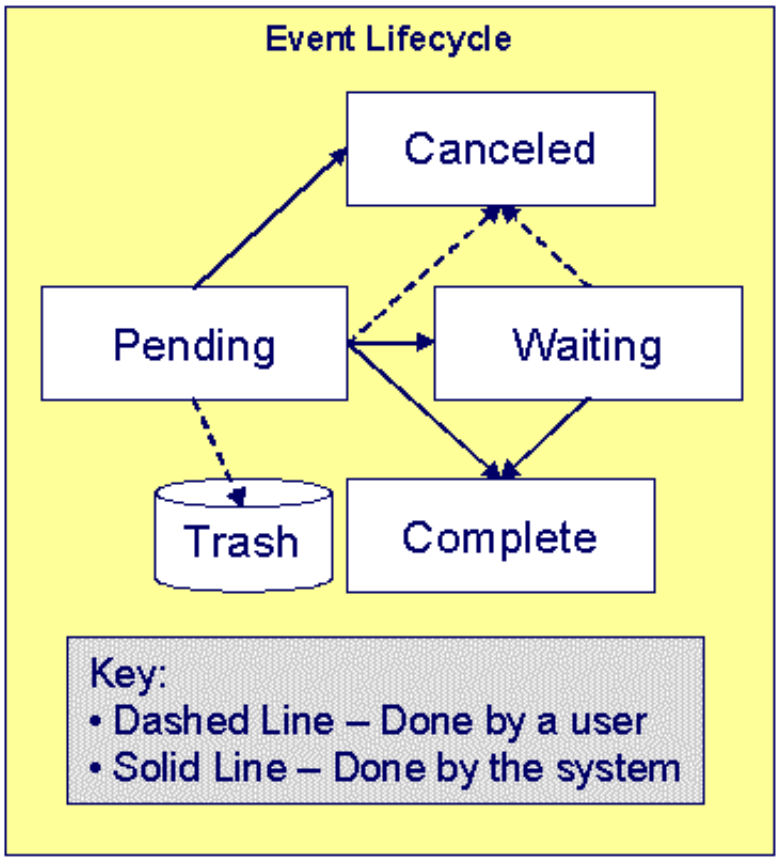
"Overdue event info" is the concatenated string of information that summarizes an overdue event throughout the system. The base-package logic constructs this string by concatenating the following information:

- The event type's description
- The event's status
- If it's `pending`:
  - If the event has a trigger date, the number of days until it's triggered plus the verbiage `day(s) from today`
  - Otherwise, the verbiage `dependent on other events`
- If it's `waiting`, the number of days, hours and minutes that it's been waiting
- If it's `canceled`, the cancel reason code's description
- If it's `complete`, the number of days, hours and minutes that it's been complete

If you'd prefer a different info string, you can develop a new algorithm and plug-it in on your event types. This design allows some / all event types to have an override info string.

Overdue Event Lifecycle

The following diagram shows the possible lifecycle of an overdue event:



Overdue events are initially created in the `Pending` state. An event can take myriad paths after it's created; it all depends on how you've configured the system. The following topics describe an event's lifecycle:

How and When Events Are Activated

An overdue event contains the date is should be activated; this is referred to as its trigger date. On this date, the Overdue / Cut Event Manager (a background process (`C1-ODET`)) invokes the `Event Activation` algorithm plugged-in on the event's event type. The `Event Activation` algorithm, in turn, will decide on the state in which to leave the overdue event (e.g., it may transition it to the `Complete` state or the `Waiting` state).

If a user can't wait for the Overdue Event Manager real-time, they can click a button on *Overdue Process - Main* to activate (and recursively trigger) events online / real-time.

You control how an event's trigger date is populated by configuring the *overdue process template*. You are given two choices when you link an event type to an overdue process template:

- You can indicate the event should be assigned a trigger date when it is first created. You'd use this approach on the first event and events with no dependencies on earlier events. The following points describe how to configure the overdue process template to do this:
  - Indicate the event type is NOT dependent on other events, and
  - Define the number of days after the process's creation to use when calculating the trigger date.

- You can indicate the event should be assigned a trigger date only after earlier events are `Complete`. This technique should be used whenever you have an event that is only executed after other events are `Complete`. The following points describe how to configure the overdue process template to do this:
  - Indicate the event is dependent on other events, and
  - Define the number of days after the completion / cancellation of all dependent event(s) that the trigger date should be set to. The Overdue Event Manager sets the trigger date on such an event when it detects that all of its dependent events are complete / canceled.

Note: Calendar vs Workdays. When an overdue event is created by the system, its trigger date is set in accordance with your date arithmetic preferences. Refer to *Calendar vs. Work Days* for more information.

Activating Events Should Add A Log Entry

As described *above*, an overdue process has a log holding a history of meaningful events in the process's life. Most `Event Activation` algorithms will add an entry to the process's log.

These log entries are more than just an audit trail as they also reference the objects that are created during activation. For example, if an activation algorithm creates a customer contact, the ID of the customer contact will be referenced on the log (and end-users will be able to drill down on the log entry to see the customer contact).

Holding Events

You can prevent a `pending` event with a trigger date on / before the current date from activating by plugging-in a `Hold Event Activation` plug-in on the overdue process template. This might prove useful, for example, if you want to suspend an overdue process while a *case* used to investigate the assertion of a customer is outstanding. Then, when the case closes, the overdue process can start up where it left off.

This technique would prove useful, for example, if your users can grant ad hoc suspend periods (e.g., if a customer wants a few extra days to pay before a cutoff). To do this, create a case type that has two states: Open and Close. Make sure to set up the Open state to have an automatic transition algorithm to close the case after X days. Then, all a user has to do is create a case of this type when they want to provide a suspend period. When the suspend period is over and payment isn't received, the case will close and the overdue process will start up where it left off.

Some Events Wait For Something Before They Activate

Consider this scenario - you want an overdue event to create a To Do entry so a user can authorize the next phase of an overdue process. When this event activates, the event's activation algorithm will create a To Do entry, but it will NOT transition the event to `complete`. Rather, the overdue event will exist in the `waiting` state. While in the `waiting` state, the Overdue Event Manager will monitor the state of the To Do entry. When the To Do entry completes, the original overdue event can transition to the `complete` state and then latter dependent events can be triggered. The following points describe how to configure the system to support this type of event:

- The event type's `Event Activation` algorithm should behave as follows:
  - It creates the object on which the overdue event waits.
  - It must link this object to the overdue process by creating a log entry where the prime-key of the related object is referenced (in a foreign-key characteristic). This log entry should also reference the event.
  - It should leave the overdue event in the `waiting` state.
- The event type must have a `Monitor Waiting Event` algorithm. This algorithm is invoked each time the *Overdue Event Manager* executes. If the related object has transitioned to a "final" state, the originating overdue event is transitioned to the `complete` state (and then latter dependent events are triggered).

Note: Bottom line. Two algorithms must be set up on an overdue event type to implement waiting functionality: an Event Activation algorithm that creates the monitored object and a `Monitor Waiting Event` algorithm to check on the state of the monitored object. The Overdue Event Manager has the dual responsibility of activating the event and monitoring its related object for completion (and then triggering the dependent events when it completes).

While the above example illustrated how an overdue event could create and then monitor a To Do entry, you can use this functionality to create and monitor any object that has an initial and final state. If the base package does not contain the algorithms you need, simply develop new ones using the base-package algorithms as examples.

How Are Events Canceled?

A `pending` event will be `cancelled` automatically by the system when the overdue process is canceled. Refer to *How Are Overdue Processes Cancelled* for more information.

A user may cancel a `pending` or `waiting` event at their discretion.

Regardless of what triggers the cancellation, the `Cancel Logic` algorithm plugged in on the overdue event type handles the cancellation. This allows you to introduce additional cancellation logic should the need arise. Please note that the base package cancel algorithms insert a *log entry* when a user manually cancels an event.

Write Offs Are Implemented Using Overdue Events

The system has been designed to allow overdue events on the original overdue process to write-off the objects being collected. Refer to *Writing Off Bills* for a case-study explaining how to do this.

Calendar vs. Work Days

When you set up your overdue process templates, you supply information that controls how each event's trigger date is calculated. You have two options:

- You can say that an event's trigger date can only be populated after earlier, dependent events are complete.
- You can say that an event's trigger date is populated when the process is first created. You simply define the number of days after the start of the process when each such event should be triggered. For example, the 2 nd event (send cutoff warning) can be triggered 7 days after the start of the process.

In addition to the above, an option defined on the *Feature Configuration for Overdue Processing* plays a part in the calculation of an event's trigger date:

- If you set the option to use `calendar days`, the trigger date of events will be set to the first workday on / after the calculated date. For example, if you indicate that the 2 nd event is triggered 7 days after the 1 st event, the system will add 7 days to the 1 st event's completion date. It then checks if this is a workday (and not a holiday); if so, this is the trigger date of the event; if not, it assigns the trigger date to the next workday.
- If you set the option to use `workdays`, the trigger date will be calculated by counting workdays. For example, if you indicate that the 2 nd event is triggered 7 days after the 1 st event, the system will count 7 workdays and set the trigger date accordingly.

Note: Account's division controls the work calendar. The system uses the above information in conjunction with the *work calendar* associated with the account's division to determine workdays.

Bill-Oriented Collection - Advanced Topics

The topics in this section provide information on features designed to facilitate the collection of overdue bills.

Miscellaneous Bill-Oriented Collection Topics

The topics in this section provide background information about a variety of bill-oriented collection topics.

Highlights Of The Sample Overdue Monitor Rule Algorithm

A base-package `Overdue Monitor Rule` algorithm exists to support many different types of overdue rules (see *C1-CB-CR-RAT*). Keep in mind that multiple such algorithms can be plugged in on *Collection Class Overdue Rules*. When the *Overdue Monitor* analyzes an account, it calls these algorithms until there are no more to invoke (or until an algorithm tells it that there is nothing more to check). The following points describe its various options:

- **Different rules based on credit ratings.** You can set up algorithms to only be applied to accounts with a credit rating <= a given value. We anticipate such algorithms will be used in conjunction with other such algorithms to apply different rules based on the customer's credit rating.
- **Different rules based on an account or contract characteristic.** You can set up algorithms to only be applied if an account or one of its contracts has a given characteristic type and value effective within the last X days. For example, you could set up an algorithm that would process an account if it had a broken payment arrangement contract within the last 365 days (broken payment arrangement contracts are marked with a given char type and value). You could set up a different algorithm that treated strategic customers more leniently (given that the definition of "strategic" could be held in an account or contract characteristic).
- **Skipping a bill if it has a future postpone date.** You can set up algorithms to ignore an unpaid bill if it has a "postpone date" that's in the future. This date would be defined on the bill using an ad hoc characteristic. The characteristic type is defined as a parameter on the algorithm.
- **Special process for bills with disputed debt.** You can set up an algorithm to create a given type of overdue process if the bill has disputed debt. By "disputed debt", we mean a bill with a financial transaction that's linked to an unbalanced match event that's marked as `Disputed`.
- **Special process for bills with credit balances.** You can set up an algorithm to create a special overdue process if the bill's total charges are negative (i.e., a credit bill). Such bills should be very unusual as we strongly recommend that credit bills should have their credit balance applied to an overpayment contract during bill completion.
- **Different rules when all contracts are inactive.** You can set up an algorithm that should only be applied if all contracts are inactive. We anticipate this can be used to differ the type of overdue process as if an account does not have active contracts; there is nothing to stop to encourage them to pay.
- **Different rules based on age and amount of a bill.** You can set up an algorithm to create different types of overdue processes based on the age and amount of the bills.

Note: You are not limited by the base-package's rules. If your implementation requires options that are not supported by the base-package algorithm, simply develop your own.

Holding A Process's Events

Refer to *Holding Events* for a description of how to prevent the activation of an overdue processes events while a given condition is true. The condition is defined in an algorithm so you can set up the system to prevent event activation for practically any condition. For example, you might want to hold event activation while the bill's account has an active case (e.g., a user might set up a case to conduct an ad hoc investigation of billing discrepancies). When the case closes, you'd want the process to start up where it left off.

Bill Info Shows Unpaid and Write-off Amounts

The base-package `Bill Information` algorithm (plugged in on the *Installation* record) is responsible for constructing a bill's summary information that appears throughout the system. This algorithm can be configured to show the unpaid and write-off amounts of each bill.

Writing Off Bills

The topics in this section provide background information on how to write off bills.

Write-Off Oriented Events

After the last contract has been stopped, overdue events to manage the write-off of the overdue process's bill(s) should be triggered. Please note that a separate overdue process is NOT created to manage write-offs. Rather, events associated with the original overdue process will handle the write-off activities.

Collection Agency Referrals

Before debt is written off, many implementations refer the unpaid bills to a collection agency. The following points describe how to implement this.

- Set up an overdue event type with an activation algorithm that refers an overdue process's bill(s) to a collection agency.
- When such an event is activated, the system creates a *Collection Referral* record for a collection agency. The specific agency to which the debt is referred is controlled by the event type's activation algorithm. The sample algorithm simply refers debt to the collection agency with the least amount of referred debt. If you prefer different logic, you must develop your own algorithm.
- A collection agency referral history record is linked to an account. It contains the amount of debt referred to the collection agency. It is the creation of this record that triggers the interface of information to the collection agency. The method used to interface the information to the agency is defined on the collection agency's record. Refer to *Setting Up Collection Agencies* for more information.
- If the collection agency is successful in obtaining the funds, a payment will be added. If the payment satisfies the cancel criteria defined on the overdue process template's cancellation plug-in, the overdue process will cancel. When an overdue process is cancelled, the cancel criteria on the overdue process's template are executed. We strongly recommend plugging in an algorithm that will cancel collection agency referrals when an overdue process is cancelled.
- If the collection agency is not successful in obtaining your funds after a given amount of time, you probably want to cancel the referral and write-off the debt. The cancellation of the referral will happen automatically if you set up your overdue process template to have an event that creates a collection agency cancellation X days after the referral. You can cancel a referral manually by simply creating a new collection agency referral history record (with a type of `cancel`).
- Collection agencies are notified of the cancellation of a referral by the creation of a new collection agency referral history record (with a type of cancel). This record will be interfaced to the agency in the same manner used to interface a new referral (see above).

Note: Log entry. The base-package overdue event activation algorithms that make and cancel collection agency referrals insert rows in the overdue process's *log* to audit these events.

Writing Off Bills Will Create Write-Off Adjustments And Possibly A Match Event

Most overdue process templates will be configured to contain an event that writes-off the unpaid balance of bills.

Note: Processing is redirected to another algorithm. It should be noted that the base-package overdue event activation algorithm that writes off bills simply redirects the call to the `Write-Off Bill` algorithm plugged in on the account's *Collection Class Overdue Rules*. The reason for this redirection is because users can manually write-off a bill (using *Bill - Main*) and when they do this, we want to invoke the same logic as when an overdue process writes-off a bill.

The base-package `Collection Class Overdue Rules - Write-Off Bill` algorithm determines the amount that should be written off for each distribution code on each unpaid financial transaction. It then creates a separate adjustment for each contract where the lines on the adjustment contain the amount to be written off for each distribution code. If the unpaid financial transactions are not linked to a match event, the write-off adjustments plus the unpaid financial transactions will be linked to a new match event. If the unpaid financial transactions were linked to an unbalanced match event, the write-off adjustments will be added to the existing match event (thus making it balanced).

If partial payments were made against a bill, the amount written off will be prorated in light of the partial payments. For example, if 20% of a bill had been paid, 80% of each distribution code will be written off.

For example, assume the original bill's FT looked as follows (note, if the bill has multiple contracts this will need to be done for each contract's FT's):

- Debit A/R $110.00
- Credit Flat Charge Revenue ($50)
- Credit Usage Revenue ($50)
- Credit City Tax Payable ($5)
- Credit State Tax Payable ($5)

Assume the customer had made a partial payment of $11 and it was matched to this FT. At write-off time, the system will create an adjustment whose adjustment lines will cause each distribution code to be written off by 90% (100% - $11 / $110). For example:

- Debit Flat Charge Revenue (or some W/O Expense) $45
- Debit Usage Revenue $45
- Debit City Tax Payable $4.50
- Debit State Tax Payable $4.50
- Credit A/R $99

This adjustment will then be linked to the original match event on which the payment was linked (thus making it `Balanced`).

Note: Log entry. The base-package overdue event activation algorithm that writes off bills inserts a row into the overdue process's *log* for each bill written off.

Small Amount Write-Downs

Many organizations will write-down a bill whose value is small early in an overdue process. The base-package overdue event activation algorithm has parameters to support this requirement (this algorithm allows you to write-off an overdue process's bill(s) if their value is less than a threshold amount).

If your organization writes-down small amounts differently than large amount, simply set up an overdue event type to reference such an activation algorithm and position it in the appropriate place in the overdue process template.

Bill Info Shows Written Off Amounts

The base-package `Bill Information` algorithm (plugged in on the *Installation* record) is responsible for constructing a bill's summary information that appears throughout the system. This algorithm can be configured to show the amount written-off for each bill.

Online Write Off Of Bills Is Performed On Bill-Main

If a bill's account is associated with a *Collection Class Overdue Rule* that has a `Write Off Bill` algorithm, a button appears on *Bill - Main* if the bill hasn't been written off. When clicked, the `Write Off Bill` algorithm is invoked to write-off the bill.

Online Reversal Of Written Off Bills Is Performed On Bill-Main

If a bill's account is associated with a *Collection Class Overdue Rule* that has a `Write Off Bill` algorithm, a button appears on *Bill - Main* if the bill has been written off. When clicked, the `Write Off Bill` algorithm is invoked (in reversal mode). The algorithm cancels the write-off adjustments (thus making the bill payable again). It also schedules the account for review by the *Overdue Monitor* the next time it runs.

Write Offs And Overdue Process Cancellation

It should be stressed that writing-off a bill may cause an overdue process to be canceled because the bill's FT will be linked to a `balanced` match event (note, the specific *cancel criteria* are in a plug-in so this is not a hard rule).

The following points highlight interesting aspects of bill write-off and overdue process cancellation:

- If a user writes off a bill *real time* and the bill has an `Active` overdue process, the process's cancel criteria will be invoked. This typically results in the cancellation of the overdue process.
- If an overdue event writes off a bill, the state of the process depends on your cancel criteria and where the overdue event is positioned in the overdue process. For example,
    - Imagine an overdue process that has an overdue event that writes offs small amounts of debt early in the process. If such a process is applied against bill with a small amount of debt, the process will be canceled (when the event activates).

- Contrast this to an overdue process where the *last* event writes off the bill. Because there are no other events to activate, the process will complete (i.e., it will not be canceled).

An Alert Can Show Written Off Bills

A base-package `Control Central Alert` algorithm (*C1-WO-BILL*) can be set up to highlight if the account in context has written off bills. This algorithm is plugged in on the *Installation* record.

Bill-Oriented Payment Arrangements

A payment arrangement is an agreement with a customer to payoff severely overdue debt in ***billed*** installments. Bills sent to customers with payment arrangements contain charges for both their current services and their payment arrangement installment amount.

Note: Nomenclature. Some organizations refer to payment arrangements as "current bill plus" agreements because the customer's bills contain charges for both their current debt plus their installment amount. After the customer has paid off their overdue debt, the payment arrangement closes and the customer's bills only contain charges for their current debt.

The topics in this section describe how to set up a payment arrangement and how the system monitors the ongoing arrangements.

Creating Payment Arrangements

When you create a payment arrangement, you are actually creating a contract. This contract is just like other contracts in that:

- It holds debt.
- It is periodically billed (thus creating unmatched bill segment financial transactions).
- When a payment is received, the payment segment financial transactions are matched to the bill segment financial transactions.

Debt is transferred to a payment arrangement contract (PA contract) from the customer's delinquent bills at the inception of the payment arrangement.

When you transfer debt from the overdue bills to a PA contract, transfer adjustments are created to transfer debt from the delinquent contracts to the PA contract. Match events are created to link the "transfer from" adjustments to the original unpaid financial transactions (FT). When all FT's on a bill are linked to `balanced` match events, the bill is no longer considered overdue and any active overdue process will be cancelled. Refer to *How Are Overdue Processes Canceled* for more information.

Note: Use the Payment Arrangement Transaction. Use the *Payment Arrangement - Bill Oriented* to set up bill-oriented payment arrangements. This transaction creates a PA contract, transfers overdue bills to it, and sets up the installment amount. This transaction is also used if you need to break or cancel the payment arrangement.

Installment, Payoff and Current Amounts

Caution: If you do not understand the difference between payoff balance and current balance, refer to *Current Amount versus Payoff Amount*.

When you set up a payment arrangement contract (PA contract), you transfer delinquent debt to the PA contract using transfer adjustments. After moneys are transferred, the system sets the PA contract's *current balance* to zero. At this point, there will be no overdue bills. If the customer neglects to pay bills containing charges associated with the payment arrangement, an overdue process will ensue.

PA contract's start their life with a non-zero *payoff balance* (i.e., they have debt when first started). This debt is transferred from the bills whose outstanding debt necessitated the creation of the PA contract.

The installment amount that the customer is billed is determined by the number of installments used to payoff the debt. For example, if the customer owes $500 and they want to pay this off in 10 installments, you'd set up the installment amount to be $50. The installment amount is saved on the PA contract's recurring charge amount. If the customer again falls into arrears on their bills, you can transfer additional bills to the PA contract. You can also change the installment amount as needed.

A PA contract's payoff balance typically differs from its current balance. The payoff balance is the amount of debt remaining to be paid off under the terms of the payment arrangement. The current balance is the installment amount that has been billed but not paid. For example, a customer who is paying off $500 with 10 installments of $50 would have an initial payoff balance of $500 and a current balance of $0. After the first bill, the PA contract would still have a payoff balance of $500, but its current balance would be $50. When the customer pays, the PA contract's payoff balance would fall to $450 and its current balance would return to $0.

The following table contains a financial example of a customer who sets up a payment arrangement to payoff $1,000 of debt in $10 installments.

Event	**Normal Contracts GL Accounting**	**PA Contract's GL Accounting**	**Normal Contract's Current Balance**	**Normal Contract's Payoff Balance**	**PA Contract's Current Balance**	**PA Contract's Payoff Balance**
Prior to creation of payment arrangement	N/A	N/A	1000	1000	N/A	N/A
Transfer debt from normal contract(s) to PA contract	Xfer 1000 A/R <1000>	PA A/R 1000 Xfer <1000>	0	0	1000	1000
Set current balance to zero on PA contract	N/A	N/A	0	0	0	1000
Customer is billed ($50 for new debt and $10 of payment arrangement debt)	A/R 50 Revenue <50>	N/A	50	50	10	1000
Customer pays $60	Cash 50 A/R <50>	Cash 10 PA A/R <10>	0	0	0	990

When the customer pays off the payment arrangement debt, the system automatically closes the PA contract after it final bills (assuming the PA contract's contract type references a bill segment type that has a bill segment creation algorithm of `Recurring Charge With Auto Stop`).

Online Canceling

A user can click a button on the *Payment Arrangement - Bill Oriented* page to cancel a payment arrangement real time. Cancellation should be used when you want to "logically delete" a PA contract because it shouldn't have been created.

The logic described above for breaking a payment arrangement is executed when a user cancels a payment arrangement; the only difference is that the PA contract is marked with a different characteristic type / value than when it is broken. The "broken" characteristic type / value can be used to apply stricter rules to the account when it's next reviewed by the *Overdue Monitor*.

Creating Overdue Procedures

Your overdue procedures define how your organization collects overdue debt. In this section, we describe how to set up the data that controls these procedures.

Caution: There are numerous ways to design your overdue procedures. Some designs will result in easy long-term maintenance; others will result in maintenance headaches. In this section, we provide information to help you understand the ramifications of the various options. Before you set up your overdue procedures, we encourage you to gain an intuitive understanding of these options by using the system to prototype the alternatives.

Designing Your Overdue Procedures

The design of your overdue procedures is an iterative process. Over time, you will develop skills that will allow you to skip some steps. However, when you're starting out, we recommend you use the following matrix as your guide. When the matrix is complete, you're ready to set up the overdue processing control tables.

Account's Collection Class **Account's Division**		

The topics discussed below will gradually complete this matrix using a simple case study.

Fastpath: For more information about how the information in this matrix is used to monitor your customers' debt, refer to *How Does The Overdue Monitor Work*.

Your Divisions Are Frequently Preordained

An account's division defines the jurisdiction whose rules govern the account. For example, an account's division controls how its payments are distributed, if / how late payment charges are levied, etc. Divisions have typically been designed in advance of designing your overdue rules.

In our example, we assume you have two divisions: North and South.

Account's Collection Class **Account's Division**		
North		
South		

Designing Your Collection Classes

Multiple collection classes are needed when your organization has different overdue rules and / or procedures based on the type of customer. If all customers are treated the same way, you'll have a single collection class (call it `Generic`). However, if you're like many organizations, you will have multiple collection classes.

For example, for commercial/industrial customers, you probably don't worry until they owe you more than, say, $100 after 20 days. For residential customers, you probably don't worry until they owe you more than, say, $5 after 20 days. In this situation, you will have at least two collection classes: one for commercial/industrial customers, the other for residential customers.

In our example, we assume you have two collection classes: Residential and Commercial/Industrial.

Account's Collection Class **Account's Division**	**Residential**	**Commercial/Industrial**
North		
South		

Note: There are two very different ways to monitor your accounts for overdue debt. This chapter describes the method referred to as Overdue Processing. Refer to *Collection / Write Off* for a description of the other method. We anticipate that most organizations will only use a single method. If your organization opts to use both methods, you will need to set up the corresponding collection classes.

Designing Overdue Monitoring Rules

At this point, we have the rows and columns defined in our matrix. Now it's time to work on the individual cells.

Note: Single currency. We've assumed that your implementation works in a single currency. If this is not true, you will need to add a 3 rd dimension that will have a value for each currency code.

Each cell will contain the rules that the *Overdue Monitor* uses to determine if an account has overdue debt. These rules will eventually be configured using one or more algorithms on *Collection Class Overdue Rules*.

Account's Collection Class **Account's Division**	**Residential**	**Commercial/Industrial**
North		
South		

Note: If the Overdue Monitor encounters an account whose collection class and division does not have overdue rules set up, it will issue an error.

Determining the rules in each cell can be straightforward or complicated; it depends on how your organization works. Our case study assumes the following:

- For residential debt (regardless of division) we have the following rules:
  - Highest priority. If a bill exists with unpaid FT's > $0 that is older than 50 days, create the "accelerated overdue process" for residential customers. We'll talk more about this process later.
  - Medium priority. If the account's has a broken payment arrangement within the last 60 days with an unpaid bill > $0 that is older than 20 days, create the "broken payment arrangement overdue process".
  - Lower priority. If a bill exists with unpaid FT's > $25 that is older than 25 days, create the "courtesy reminder overdue process" for residential customers. We'll talk more about this overdue process later.
- For commercial-industrial debt (regardless of division) we have the following rules:
  - Highest priority. If a bill exists with unpaid FT's > $0 that is older than 45 days, create the "commercial 45 days late overdue process". We'll talk more about this process later.
  - Medium priority. If the account's credit rating is < 550 and has an unpaid bill > $0 that is older than 20 days, create the "risky commercial customer overdue process".

- Lower priority. If a bill exists with unpaid FT's > $100 that is older than 30 days, create the "commercial 30 days late overdue process". We'll talk more about this overdue process later.

Given the above, our matrix will look as follows:

Account's Collection Class **Account's Division**	**Residential**	**Commercial/Industrial**
North	Highest Priority: If a bill exists with unpaid FT's > $0 that is older than 50 days, create the accelerated overdue process for residential customers. Medium Priority: If the account has a broken payment arrangement within the last 60 days with an unpaid bill > $0 that is older than 20 days, create the broken payment arrangement overdue process. Lowest Priority: If a bill exists with unpaid FT's > $25 that is older than 25 days, create the courtesy reminder overdue process for residential customers.	Highest Priority: If a bill exists with unpaid FT's > $0 that is older than 45 days, create the commercial 45 days late overdue process. Medium priority. If the account's credit rating is < 550 and has an unpaid bill > $0 that is older than 20 days, create the risky commercial customer overdue process. Lowest Priority: If a bill exists with unpaid FT's > $100 that is older than 30 days, create the commercial 30 days late overdue process.
South	Same as above	Same as above

Note: The rules are limited by your imagination (and business requirements). While the base-package `Overdue Monitor Rule` algorithm (*C1-CB-CR-RAT*) supports the above scenarios, we'd like to stress these are just examples. Your implementation can operate using very different rules by either configuring the base-package algorithm (it has many parameters that you can use to tailor your rules) OR by introducing a new algorithm. Refer to *Highlights Of The Sample Overdue Monitor Rule Algorithm* for the options delivered with the base-package.

Designing Overdue Process Templates and Event Types

The following table shows a sample overdue process template for one of the rules in the Residential / North cell in the previous section's matrix.

Overdue Process Template	**Overdue Event Type**	**When Triggered**
Accelerated overdue process for residential customers	Old debt letter	At inception of process
	Reduce customer's credit rating	10 days after inception
	Write down small debt	0 days after the set event

Overdue Process Template	Overdue Event Type	When Triggered
	Refer debt to collection agent	0 days after attempting the small write down (this means that either the small write-down or the agency referral will take place as if the write-down is successful, the bill's FTs will be matched to balanced match events and the overdue process will stop)
	Cancel collection agent referral	45 days after referral
	Write-off debt	0 days after collection agent cancellation

You should create a similar table for each of the distinct overdue process templates in your matrix.

At this point, you've designed the distinct overdue process templates. Next, you'll need to design the algorithms that control their overdue processes:

- A template's `Calculate Unpaid and Original Amount` algorithm calculates the original and unpaid amounts of the objects being collected by a process. These values are used throughout the overdue processing module.
- A template's `Cancel Criteria` algorithm is executed to determine if a process should be cancelled. Refer to *How Are Overdue Processes Cancelled* for the details.
- A template's `Cancel Logic` algorithm is executed to cancel a process. Refer to *How Are Overdue Processes Cancelled* for the details. Please note that the logic embodied in this type of algorithm can be sophisticated because it is responsible for stopping an ongoing process's activities. Cancellation algorithms are also responsible for inserting *log* entry(s).
- A template's `Hold Event Activation` algorithm is invoked to determine if the *Overdue Event Manager* should *suspend the activation* of the process's events.
- A template's `Information` algorithm is invoked to construct the *override information string*.

Next, extract each unique event type from the above table:

Overdue Event Type	Action
Old debt letter	Create a customer contact
Reduce customer's credit rating	Insert an account credit rating history record
Write down small debt	Create write-down adjustments if unpaid debt is less than $x
Refer debt to collection agent	Create a collection agency referral
Cancel collection agent referral	Cancel the collection agency referral
Write off unpaid debt	Create adjustments to write-off unpaid debt

At this point, you know the distinct event types. Next, you'll need to design the algorithms that control the lifecycle of each event type:

- The event type's `Event Activation` algorithm(s) are executed by the *Overdue / Event Manager* on its trigger date. The following points describe the logic embodied in such an algorithm:
    - The activity that happens on the trigger date (e.g., creation of a customer contact, To Do, etc.). Refer to *Overdue Events Can Do Many Things* for the details.

    - Whether the event is transitioned into the `Waiting` or `Complete` state when it's triggered. Refer to *Some Events Can Wait* for the details.
    - How the log entry(s) associated with event activation will be constructed. The base-package algorithms allow you to control the verbiage in the log entry by defining the desired message number on the algorithm. This means that you may have to set up new messages. Refer to *Activating Events Should Add A Log Entry* for the details.
- The event type's `Cancel Logic` algorithm(s) are invoked when *an event is cancelled*. The following points describe the logic embodied in such an algorithm:
    - If the event is allowed to be canceled. This logic may be necessary if some conditions prevent events of this type from canceling. For example, you may want to prevent an event from canceling when there are later dependent events that aren't canceled.
    - Any ancillary actions that take place during cancellation.
    - How the *log entry(s)* associated with event cancellation will be constructed.
- The event type's `Monitor Waiting Event` algorithm(s) are invoked to *monitor a waiting event*. These algorithms are responsible for transitioning a `Waiting` event to `Complete` if the object on which it's waiting is complete.
- The event type's `Event Information` algorithm is invoked to construct the *override "info string"*.

Set Up Tasks

The above topics provided background information about how overdue processing works. The following discussion summarizes the various set up tasks.

Overdue Event Types

You will find that most of the time spent setting up your overdue event types is spent setting up the objects that are referenced on the overdue event type algorithms. For example, if you use the base-package algorithms, you will set up the following:

- The various "types" for the objects created by the plug-ins. For example,
    - If an overdue event type creates a To Do entry, you must set up the To Do type.
    - If an overdue event type creates a customer contact, you must set up the customer contact type.
    - If an overdue event type writes off debt, you must set up the adjustment types.
    - ...
- *Foreign key characteristic types* that are used to reference the ancillary objects in the *log entries* (e.g., if an event creates a customer contact, the log references this customer contact using a FK characteristic type). Note, many of these will exist in the base-package.
- *Messages* that are used to define the verbiage in the *log entries*. For example, if you use the base-package algorithm that creates a customer contact, you must supply the desired message category and number that contains the verbiage that appears in the log when customer contacts are created. Note, messages have been set up for all base-package algorithms (this means you should not have to set up new messages).
- Etc.

The only way to compile the complete list is to design the parameters for each overdue event type algorithm. Refer to *Overdue Event Type - Main* for the supported plug-in spots.

After you've set up the objects referenced on the algorithms, you can then set up the algorithms. Only then can you set up the overdue event types.

Overdue Process Templates

After your overdue event types exist, you can set up your overdue process templates. You will find that most of the time spent setting up your overdue process templates is spent setting up the objects that are referenced on the overdue process template algorithms. Refer to *Overdue Process Template - Main* on page 639 for the supported system events.

Collection Classes

Set up *collection classes* as per your *overdue procedures*. Make sure to indicate that these collection classes use the `Overdue` collection method (only accounts linked to collection classes designated as using the `Overdue` collection method or processed by the Overdue Monitor).

Collection Class Overdue Monitor Rules

After your overdue process templates exist, you can set up your `Overdue Monitor Rules`. These rules are algorithms plugged in on *Collection Class Overdue Rules*. You will find most of the time spent setting up these algorithms is spent setting up the objects referenced on the base-package algorithm.

Feature Configuration

You must set up a *Feature Configuration* to define parameters that control various overdue processing options.

The following points describe the various **Option Types** that must be defined:

- `Trigger Date: Y-Workdays, N-Calendar Days.` This option controls how the system computes the trigger dates on overdue events. Enter `Y` if the system should use workdays. Enter `N` if the system should use calendar days. Refer to *Calendar vs Work Days* for the details.
- `Payment Arrangement Type (B/S/A).` This option indicates whether your implementation uses the *Balance-Oriented Payment Arrangements* (value ***S***), *Bill-oriented Payment Arrangements* (value ***B***) or both (value ***A***).
- `Champion Template$Challenger Template$Percentage(1-100).` You need only set up options of this type if your implementation implements *Champion / Challenger* functionality. Options of this type are entered in the format `A$B$nnn` where A is the overdue process template of the champion template, B is the overdue process template of the challenger template, and C is the percent of the time that the system should create the challenger template. The overdue monitor uses this option to override the champion overdue process template X% of the time with the challenger template. You may enter any number of these options (but only one per Champion Template).

Overdue Event Cancellation Reasons

Overdue events can be cancelled automatically and manually (at the discretion of a user). Regardless of the method of cancellation, a cancellation reason must be supplied. You set up your overdue event cancellation reasons using *Overdue Event Cancellation Reason - Main*.

Collection Agencies

If you refer debt to collection agents, you must set up your *collection agencies*.

Alert To Highlight Active Overdue Processes

If you want an alert to appear if the account has active overdue processes, you must configure an appropriate `Control Central Alert` algorithm (*C1-OD-PROC*). This algorithm is plugged in on the *Installation* record.

Bill-Oriented Collection - Additional Set Up

The topics in this section provide information on additional set up requirements if you collect on unpaid bills.

- *One Bill Per Match Event* on page 636
- *Bill-Based Payment Arrangements* on page 636
- *Bill-Based Write-off* on page 636

- *Alert To Highlight Written Off Bills* on page 637
- *Open-Item Bill Amount Plug-In* on page 637

One Bill Per Match Event

As mentioned earlier, a bill is considered paid if its financial transactions (FTs) are linked to a `balanced` match event. To determine a bill's outstanding amount, FTs from different bills cannot be commingled on the same match event (but it's OK for a bill's FTs to be on multiple match events). If you stick by the rule of "just one bill per match event" you will then be able to determine the outstanding balance of a partially paid bill. However, if you mix more than one bill under a match event, a particular bill's balance may become indeterminate.

The following Open-Item algorithm types have been provided by the base package to help enforce this rule:

- The Distribute by Bill Due Date `Payment Distribution` algorithm (*C1-PYDS-BDU*).
- The match by Bill ID `Payment Distribution Override` algorithm (*C1-PDOV-PYBL*).
- The FT cancellation `FT Freeze` algorithm (*C1-CFTZ-COFT*).

If any of your customized plug-ins and processes create match events, it is important that these too enforce this rule. You may want to refer to the base package algorithms as an example of how to do this.

Bill-Based Payment Arrangements

If you set up *payment arrangements for unpaid bills*, you must configure the `Bill-Based Payment Arrangement` algorithm and plug it in on your *Collection Class Overdue Rules*.

It's important to set up the adjustment types used during payment arrangement creation to NOT print on bills. This is because the base-package algorithm will match the adjustments used to transfer debt to the payment arrangement with the adjustment used to reduce the payment arrangement's current amount by the amount of the transfer if all adjustment types are set up to not print.

Bill-Based Write-off

If you *write-off unpaid bills*, you must set up the following:

- Set up the adjustment type that will be used to write-off an unpaid financial transaction. This adjustment type must be configured as follows:
  - **Adjustment Amount Type** must be `Calculated Amount`
  - Its distribution code is irrelevant as a separate calculation line will be created for each distribution code on the FT's that is written off and these lines will reference the appropriate distribution code.
  - It must reference an *adjustment type* char type / value that identifies it as one used to write-off a bill's FTs
  - The following algorithms must be defined on the adjustment type:
- The `Generate Adjustment` system event must reference an algorithm that has the responsibility of determining how to write-off a FT. This algorithm should be determining the FT's GL details and creating a separate adjustment calculation line for each GL detail. The base package is supplied with a sample algorithm that does this (*C1-ADJG-WO*).
- The `Adjustment Financial Transaction` system event should reference an algorithm that impacts current, payoff and the GL by the amount being written off.
- The distribution codes referenced on the financial transactions must be set up with a characteristic that holds the distribution code used to write-off the original amounts. For example,
  - Distribution codes used to record tax liabilities will typically reference the same distribution code for write-off (most organizations *reverse* tax liabilities at write-off time)
  - Distribution codes used to record revenue will typically reference a write-off distribution code used to record a write-off expense.
  - Distribution codes used to record receivables will typically not reference a write-off distribution code because receivables are implicitly written off when revenue and tax liabilities are written off.

- Set up an adjustment cancellation code used when a users reverses a written-off bill (reversal involves canceling the write off adjustments).
- Set up a `Write Off Bill` algorithm and plug it in on your *Collection Class Overdue Rules*. This algorithm will reference the adjustment type described in the previous point.

Alert To Highlight Written Off Bills

If you want an alert to appear if the account has bills with written-off debt, you must configure an appropriate `Control Central Alert` algorithm (*C1-WO-BILL*). This algorithm is plugged in on the *Installation* record.

Open-Item Bill Amount Plug-In

You must set up the algorithm that computes the original, unpaid, and write-off amounts of your open-item bills. This algorithm is called by other algorithms when these amounts are needed. This algorithm is plugged-in on *Installation* in the `Determine Open Item Bill Amounts` spot.

Setting Up Overdue Processing

The topics in this section describe how to set up the control tables to implement your overdue processing.

Setting Up Overdue Event Types

An overdue event type encapsulates the business rules that govern a given type of overdue event. Open **Admin Menu, Overdue Event Type** to set up overdue event types.

Note: Recommendation. Before using this transaction, we strongly recommend that you review *The Big Picture Of Overdue Events*.

Description of Page

Enter a unique **Overdue Event Type** code and **Description** for the overdue event type.

Use **Long Description** to provide a more detailed explanation of the purpose of the overdue event type.

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

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to `10` unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Optional / Required	Description
`Cancel Logic`	Required	This algorithm is executed to cancel an overdue event. Refer to *How Are Events Canceled* for the details. Click *here* to see the algorithm types available for this system event.
`Event Activation`	Required	This algorithm is executed to activate an overdue event on its trigger date. Refer to *Overdue Events Can Do Many Things* and *How and When Events Are Activated* for the details. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Event Information`	Optional - only used if you want to override an overdue event's info string	This algorithm is executed to construct an overdue event's override info string. Refer to *Overdue Event Information Is Overridable* for the details. Click *here* to see the algorithm types available for this system event.
`Monitor Waiting Events`	Optional - only used if events of this type can enter the `Waiting` state	This algorithm is invoked by the Overdue Event Manager for events in the Waiting state. Refer to *Some Events Wait For Something Before Completing* for the details. Click *here* to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_OD_EVT_TYPE*.

Setting Up Collection Class Overdue Rules

Collection class overdue rules contain algorithms that impact accounts associated with a given collection class, division and currency code are managed. Open **Admin Menu, Collection Class Overdue Rules** to set up collection class overdue rules.

Note: Recommendation. Before using this transaction, we strongly recommend that you review *Different Overdue Rules For Different Customers*.

Description of Page

Enter the **Collection Class**, **Division** and **Currency Code** to which the rules apply.

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

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to `10` unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

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

System Event	Optional / Required	Description
`Bill-Based Payment Arrangement`	Optional - only specified if your implementation uses bill-oriented payment arrangements	This algorithm is executed to handle the creation, breaking and canceling of a *Bill-Oriented Payment Arrangements*. Click *here* to see the algorithm types available for this system event.

System Event	Optional / Required	Description
`Overdue Monitor Rule`	Required	This algorithm is invoked by the Overdue Monitor to analyze an account's debt. Refer to *How Does The Overdue Monitor Work* for the details. If you have multiple rules (and therefore multiple algorithms), please take care when assigning the sequence number, as the Overdue Monitor will invoke these rules in sequence order. Click *here* to see the algorithm types available for this system event.
`Write Off Bill`	Option - only specified if your implementation writes-off bills	This algorithm is executed to handle the write-off and write-off reversal of a bill. Refer to *Writing Off Bills*. Click *here* to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_OD_RULE_ALG*.

Setting Up Overdue Event Cancellation Reasons

An overdue event cancel reason must be supplied before an overdue event can be canceled. Open **Admin Menu, Overdue Event Cancel Reason** to define overdue event cancellation reasons.

Description of Page

Enter an easily recognizable **Overdue Event Cancel Reason** and **Description** for each cancellation reason.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_OEVT_CAN_RSN*.

Overdue Process Template Maintenance

An overdue process template encapsulates the business rules that govern a given type of overdue process. Open **Admin Menu**, **Overdue Process Template** to set up overdue process templates.

Note: Recommendation. Before using this transaction, we strongly recommend that you review *The Big Picture Of Overdue Processes*.

Overdue Process Template - Main

Description of Page

Enter a unique **Overdue Process Template** and **Description** for the overdue process template.

Collecting On Object defines the type of object managed by this overdue process. This field actually references a foreign key characteristic type that references the managed object. For example, if this overdue process template manages overdue bills, you'd reference a foreign key characteristic that references the bill object.

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

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).

- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to `10` unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	Optional / Required	Description
`Calculate Unpaid & Original Amount`	Required	This algorithm is executed to calculate the unpaid and original amounts of the objects associated with the overdue process. These amounts are shown on the overdue process page and in the base-package *overdue info string*. Click *here* to see the algorithm types available for this system event.
`Cancel Criteria`	Required	This algorithm is executed to determine if an overdue process can be cancelled. Refer to *How Are Overdue Processes Cancelled* for the details. Click *here* to see the algorithm types available for this system event.
`Cancel Logic`	Required	This algorithm is executed to cancel an overdue process. Refer to *How Are Overdue Processes Cancelled* for the details. Click *here* to see the algorithm types available for this system event.
`Hold Event Activation Criteria`	Optional - only used if overdue processes of this type can be suspended while some condition is true	This algorithm is executed to determine if the activation of overdue events should be suspended. Refer to *Holding Events* for the details. Click *here* to see the algorithm types available for this system event.
`Overdue Process Information`	Optional - only used if you want to override an overdue process's info string	This algorithm is executed to construct an overdue process's override info string. Refer to *Overdue Process Information Is Overridable* for the details. Click *here* to see the algorithm types available for this system event.

The **Event Types** control the number and type of overdue events linked to an overdue process when it is first created. The information in the scroll defines these events and the date on which they will be triggered. The following fields are required for each event type:

- **Event Sequence.** Sequence controls the order in which the overdue event types appear in the scroll.
- **Overdue Event Type.** Specify the type of overdue event to be created.
- **Days After.** If **Dependent on Other Events** is on, events will be triggered this many days after the completion of the dependent events (specified in the grid). Set this value to 0 (zero) if you want the event triggered immediately after the completion of the dependent events. If **Dependent on Other Events** is off, events will be triggered this many days after the creation of the overdue process. Refer to *How and When Events Are Activated* for the details.
- If **Dependent on Other Events** is on, define the events that must be completed or cancelled before the event will be triggered.

- **Sequence** is system-assigned and cannot be specified or changed.
- **Dependent on Sequence** is the sequence of the dependent event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_OD_PROC_TMP*.

Overdue Process Template - Characteristics

The **Characteristics** tab allows you to define characteristics for an overdue process template. It contains the following fields in a grid:

Field Name	Field Description
Sequence	Used to specify the sequence number for the characteristic. The sequence number must be unique when you want to define multiple characteristics using the same characteristic type.
Characteristic Type	Used to indicate the characteristic type. **Note:** The list includes only those characteristic types where the characteristic entity is set to **Overdue Process Template**.
Characteristic Value	Used to specify the value for the characteristic type. **Note:** When you specify the value for a predefined characteristic type, the description of the characteristic value appears corresponding to the **Characteristic Value** field. When you select a predefined characteristic type, the **Search** icon appears corresponding to the **Characteristic Value** field. On clicking the **Search** icon, you can search for a predefined characteristic value.

If you want to define more than one characteristic for the overdue process template, click the **Add** ( ) icon and then specify the details. However, if you want to remove a characteristic from the overdue process template, click the **Delete** ( ) icon corresponding to the characteristic.

Chapter 17

Defining Batch Schedule Options

Topics:

The topics in this section describe how to set up the system to periodically execute batch job streams.

Note: The batch scheduler is optional. Setting up the batch scheduler is only necessary if your organization uses the product's batch scheduler. If your organization uses a third party tool to manage the periodic execution of batch jobs, this section is not applicable.

Note: Separate module. Please note that batch scheduler functionalityis associated with separate `Workflow Scheduling` module. If this module is not applicable to your business you may turn it off. Refer to *Turn Off A Function Module* for more information.

The Big Picture of Scheduling Batch Jobs

The following points provide an overview of how batch jobs are scheduled to run periodically.

Job Stream, a Definition

The term "job stream" refers to a suite of batch jobs that run periodically. Most organizations have multiple job streams. For example,

- You'll have a job stream that contains batch jobs that run nightly
- You'll have a different job stream that contains the hourly batch jobs
- You'll have a different job stream that contains the batch jobs that extract data for your data warehouse
- Etc.

Fastpath: Refer to *Batch Process Dependencies* for a description of sample job streams.

A Workflow Process Is Created Each Time A Job Stream Executes

The system creates a *workflow process* each time a job stream executes. The workflow process has a separate workflow event for each batch job in the job stream. The batch jobs are submitted when the workflow process's events are activated (i.e., each workflow process event submits a specific batch job).

A Workflow Process Template Defines The Batch Jobs In A Job Stream

The system creates a job stream's workflow process using a *workflow process template*. This means that a separate workflow process template exists for each job stream. For example, there is a workflow process template for the nightly job stream and another for the weekly job stream, etc.

There are typically dependencies between a job stream's batch jobs. For example, the billing batch job might be dependent on the successful execution of the payment upload batch job. Dependencies between batch jobs are defined by setting up workflow event dependencies on the workflow process. For example, if the billing batch job should only run after the payment upload batch job completes, you'd set up the workflow event that submits the billing job to be dependent on the completion of the event that submits the payment upload job. Note, you can define multiple dependencies between batch jobs.

Once you define your job streams using workflow process templates, you indicate your job streams in the `batch scheduler` *feature configuration*. The *job stream summary* page and the *job stream creation schedule* use this information to display the appropriate job stream templates.

Note: Importing workflow process templates from the demonstration database. Refer to *How To Copy Job Streams From The Demonstration Database* for a description of how to copy sample workflow process templates from the demonstration database.

How To Start A Job Stream

You can manually start a job stream using the *Job Stream Summary* page.

Optionally, you can *set up a schedule* defining when the system should start a job stream (i.e., create a workflow process). For example, you can set up a schedule for the "nightly" workflow process template to indicate that a workflow process should be created every night at 5:30 pm, Monday through Friday.

- You can monitor the status of your job streams on the *Job Stream Summary* page.

- You can monitor the status of a specific execution of a job stream on the *Job Stream Details* page. This page shows the status of the events on a workflow process and a summary of the execution status of each event's batch job.

Fastpath: Refer to *Technical Implementation Of The Batch Scheduler* for more information.

Activating A Workflow Event Causes A Batch Run To Be Submitted

When a workflow event is activated, its activation algorithm creates a `Pending` batch run request. To be specific - the workflow event's `Activation` algorithm defines the batch process and the *number of parallel threads* that should be executed when the batch job is submitted. Please see the activation algorithm's parameters for a description of exactly how the `Pending` batch run request is created.

Note: What activates workflow events? A batch process exists that is responsible for activating the workflow events associated with your job streams' workflow processes. This batch process is known as *C1-WFSUB* . It should be noted that when this process is submitted, it only activates the workflow events associated with a specific workflow process template (the workflow process template is identified by a parameter supplied to `C1-WFSUB`). You may wonder why the standard `WFET` batch process (i.e., the standard workflow event activator) doesn't process the batch scheduler workflow events. The reason is that `C1-WFSUB` is executed many times during the processing of your job streams and we didn't want to clutter up the run statistics for `WFET` with the myriad executions of `C1-WFSUB`.

Workflow Event Status Reflects The Status Of The Batch Run

The system creates a *workflow process* each time a job stream executes. The workflow process has a separate workflow event for each batch job in the job stream. The batch jobs are submitted when the workflow process's events are activated (i.e., each workflow process event submits a specific batch job). The following diagram shows the potential state of these workflow events:

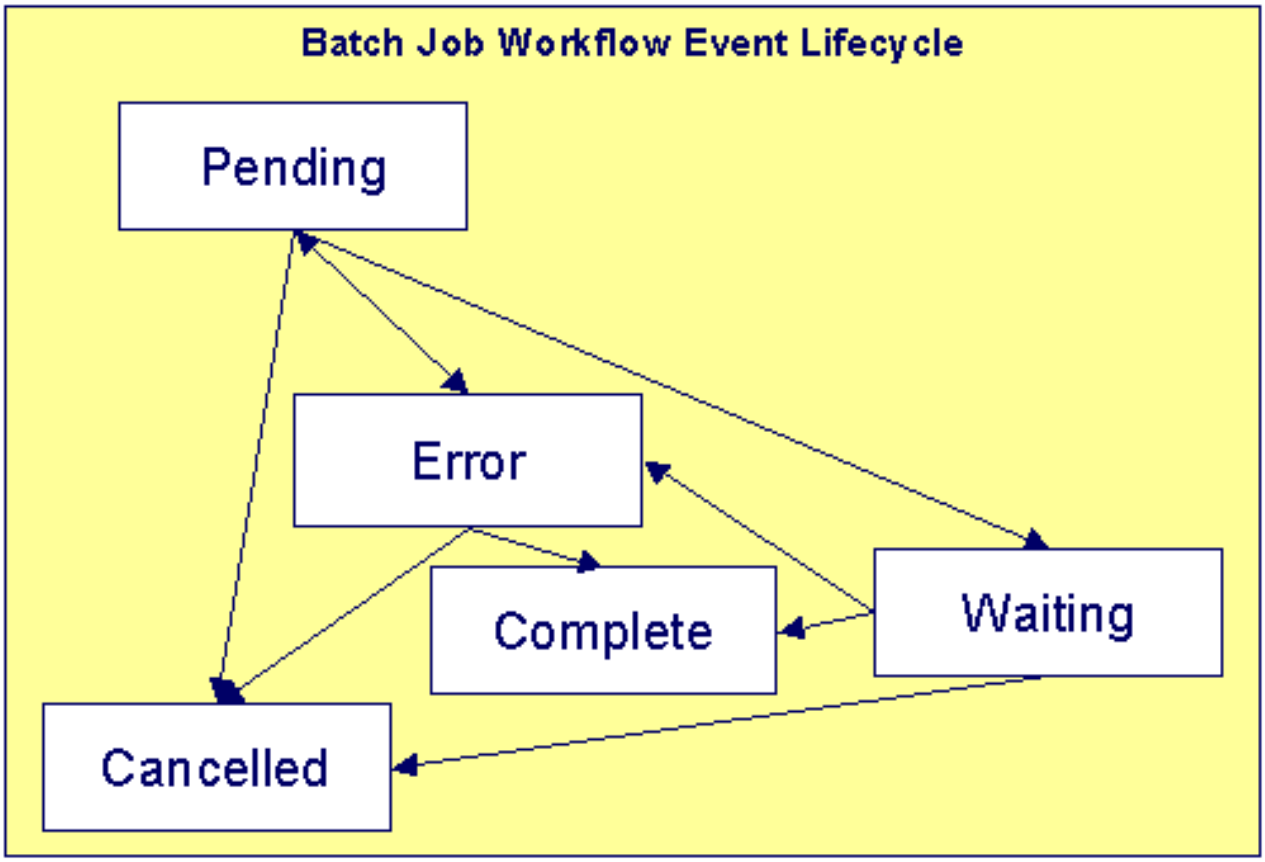


Figure 6: Workflow Event Lifecycle

The following points explain the relationship between a workflow event's status and the state of the corresponding batch job that it submits:

- Workflow events are initially created in the `Pending` state. The event's batch job has not been submitted when it's in this state.
- When the workflow event is activated, its activation algorithm attempts to submit a request to execute a batch job:
    - If there is something wrong with the activation algorithm's parameters, the event will enter the `Error` state. If this happens, you can:
- Resubmit the batch job by changing the event's state back to `Pending`.
- Cancel the batch job by changing the event's status to `Canceled`.
- Skip this batch job by changing the event's status to `Complete`. Do this if the subsequent dependent batch jobs should proceed despite these errors.
    - If the batch run is submitted successfully, the workflow event enters the `Waiting` state (it is waiting for the batch job to complete).
    - When the batch job completes, the workflow event transitions into either the `Complete` or `Error` state:
        - If the batch run aborts due to too many errors, it transitions into the `Error` state. If this happens, you can:
    - Restart a batch job that aborted by changing the event's status back to `Pending`.
    - Cancel the batch job by changing the event's status to `Canceled`.
    - Skip this batch job by changing the event's status to `Complete`. Do this if the subsequent dependent batch jobs should proceed despite these errors.

- If the batch run doesn't abort, due to too many errors, it transitions to the `Complete` state.
- You can `Cancel` a `Pending` event if you don't want the batch job to be submitted.
- A `Pending` event will be `Canceled` automatically by the system if the workflow process is canceled by a user.
- A `Waiting` event will be `Canceled` automatically by the system if the workflow process is canceled by a user.

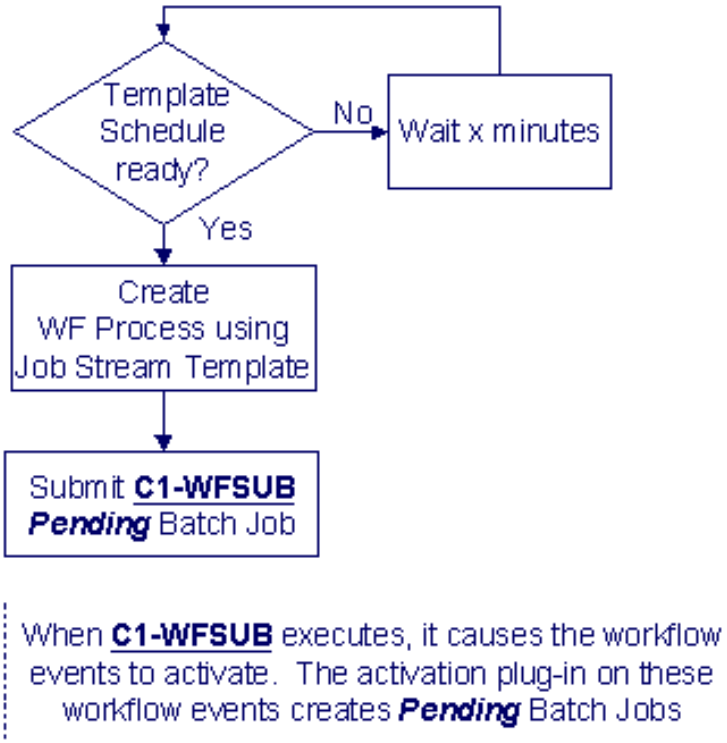
Note: You can monitor the status of a workflow process's events and their related batch jobs on the *Job Stream Details* page.

Technical Implementation Of The Batch Scheduler

The topics in this section provide information about how your job streams are executed. This section is intended for your technical personnel who are responsible for configuring the program that periodically submits batch jobs.

A PERL Program Determines If There's Work To Do

A PERL program runs in the background looking for job stream (workflow process) templates whose *schedule* is ready to be processed. The following flowchart provides a schematic of its logic:



The following points summarize important concepts illustrated in the flowchart:

- When the job stream template *schedule* indicates it's time to start a job stream, a workflow process is created by copying the event types from the job stream (workflow process) template.
- In addition, a `Pending` batch job that activates the workflow process's events is created (i.e., the `C1-WFSUB` batch job is submitted).
- When `C1-WFSUB` executes, it activates the workflow process's events. The activation plug-in of these workflow events creates the job stream's `Pending` batch jobs. In addition, the activation algorithm transitions the workflow events into the `Waiting` state (they are waiting for the batch job to complete).

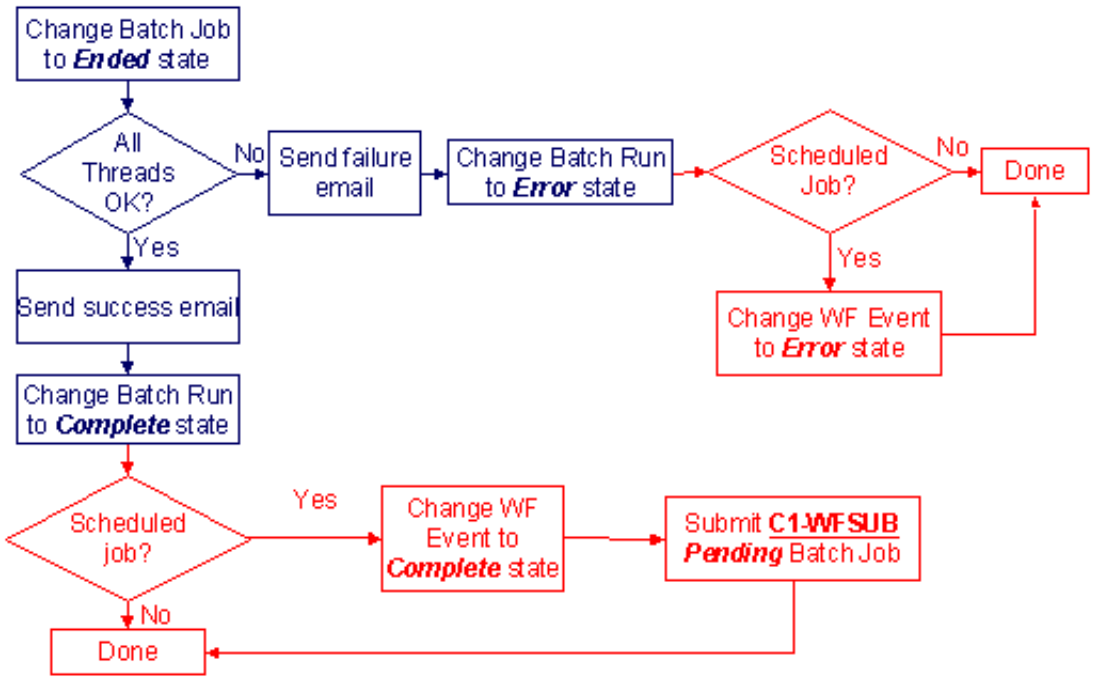
Note: The identity of `C1-WFSUB` is not hard-coded. Rather, it's defined on the batch scheduler's *Feature Configuration*.

The same PERL program described above looks for `Pending` batch jobs and then executes them for the standard batch job submission functionality.

Note: Number of Threads. For batch jobs related to a job stream, the number of parallel threads to execute is defined on a parameter on the workflow event's activation algorithm.

Completing A Batch Program That's Part Of A Job Stream

When a batch program completes, a common routine is called. The following flowchart provides a schematic of this routine's logic where the additional logic applicable to the batch scheduler is highlighted in red.



The following points summarize important concepts regarding the batch scheduler logic illustrated in the flowchart:

- If at least one thread fails and this batch job is one related to a scheduled job stream, the status of the corresponding workflow event is changed to `Error`.
- If all threads are successful, and this batch job is one related to a scheduled job stream,
  - The status of the corresponding workflow event is changed to `Complete`.
  - In addition, a `Pending` batch job is submitted to activate workflow events that are dependent on the completion of a batch job. When this job executes, the "downstream" batch jobs will be submitted (and then the logic shown above starts again).

Note: Batch Run Number. The routine also creates a characteristic for the workflow event to capture the batch run number associated with this batch run. The characteristic type to use is defined on the *feature configuration*.

Setting Up The Batch Scheduler

The following topics summarize how to set up the system to enable the scheduling of job streams.

Create A Workflow Event Type - Activation Algorithm For Every Batch Job

As described under *The Big Picture of Scheduling Batch Jobs*, the system creates a workflow process when a job stream is initiated. The workflow process's events cause `Pending` batch jobs to be created when the workflow events are activated..

A base-package Workflow Event Type - Activation algorithm exists that creates a `Pending` batch job (the identify of the batch job is defined as a parameter on the algorithm). This means that you need to set up a `Workflow Event Type - Activation Algorithm` for every batch job.

Besides defining the batch job's *batch control* code, the base-package algorithm, *C1-WFAC-CRBJ* also allows you to define the following values on each algorithm:

- **The number of parallel threads that should be submitted.** If this parameter is left blank, the batch job is submitted using a single thread. If a value greater than one is defined, the system automatically initiates this number of parallel threads. For example, if you want to kick the `BILLING` batch process using 20 parallel threads, specify a value of 20 in the respective algorithm's parameters. Refer to *Optimal Thread Count for Parallel Background Processes* for more information of executing a batch process in parallel threads.
- **The override email address used to inform that a batch job has finished.** If this parameter is specified, email is sent to this address when a batch job completes informing the recipient of its successful completion (or failure). Note, this address overrides the email address defined on the *Feature Configuration* for the batch scheduler (meaning that you need only populate this parameter if you want the email to be sent to someone other than the global email recipient defined on the Feature Configuration).
- **The value of all batch job-specific parameters.** Some batch jobs have job-specific parameters. For example, the Account Debt Monitor (ADM) batch process has a parameter that controls if calendar or workdays should be used when calculating dates. Values for the job-specific parameters may be defined on the batch code. In addition, override values may be defined on the algorithm's parameters. For example, you can define if calendar days or workdays are used when you set up the algorithm that submits the Account Debt Monitor batch process.

Note: Importing saves time. If you *import job streams* from the demonstration database, the import process will set up a separate Workflow Event Type - Activation Algorithm for every batch job in the system. You need only add additional algorithms if your implementation develops new batch processes.

Note: Demonstration data and parameter values. Please note that the workflow event type activation algorithms provided in the demonstration database do not define any job-specific parameter values. In addition, most batch control records in the demonstration database do not define default job-specific parameter values. Your organization must determine the appropriate values for your business and define these values accordingly.

Create A Workflow Event Type For Every Batch Job

As described under *The Big Picture of Scheduling Batch Jobs*, the system creates a workflow process when a job stream is initiated. The workflow process's events reference workflow event types. A separate workflow event type must be set up for every batch job (and you must reference the appropriate activation algorithm on each workflow event type).

Note: Importing saves time. If you *import job streams* from the demonstration database, the import process will set up a separate Workflow Event Type for every batch job in the system. You need only add additional workflow event types if your implementation develops new batch processes.

Create A Workflow Process Template For Every Job Stream

As described under *The Big Picture of Scheduling Batch Jobs*, the system creates a workflow process when a job stream is initiated. The system uses a workflow process template to create a workflow process. This means you must create a workflow process template for every batch job.

> **Note: Importing saves time.** If you *import job streams* from the demonstration database, the import process will set up a workflow process templates for the classic job streams used in the demonstration database. We recommend that you modify imported workflow process templates to only include workflow event types for the batch jobs applicable to your implementation (the workflow process templates in the demonstration database contain a workflow event type for every possible batch job and it is unlikely that your implementation uses every possible batch job).

Set Up A Work Calendar

As described under *How To Start A Job Stream*, you must set up a Job Stream Creation Schedule to define when each job stream should be started. You do this using the *Job Stream Creation Schedule* transaction. This transaction allows you to create the submission dates and times using a "recurrence schedule" (i.e., you don't have to type in 52 entries for a weekly job stream, you can rather have the system create the 52 entries for you). This feature uses a given *work calendar* to know what days are workdays and holidays. You must set up a work calendar (or reuse an existing work calendar) if you intend to use this feature.

Set Up A Job Stream Creation Schedule For Each Job Stream

As described under *How To Start A Job Stream*, you must set up a *Job Stream Creation Schedule* to define when each job stream should be initiated. For example, you can set up a schedule to have your "nightly" job stream run at 5pm, Monday through Friday.

Set Up Characteristic Types

When the system activates a workflow event that submits a `Pending` batch job, it updates the workflow event with information about the batch job. It does this by populating characteristics on the workflow event. In order for the system to do this, you must set up the following characteristic types:

- Batch Job ID. This must be FK characteristic to the Batch Job object.
- Batch Control Code. This must be a FK characteristic to the Batch Control object.
- Batch Run Number. This must be an ad hoc characteristic.

In addition to the above workflow event characteristics, you must also set up a workflow process characteristic:

- Business Date. This must be an ad hoc characteristic. It holds the business date that's passed to the job stream's batch jobs (all batch jobs in a job stream use the same business date).

Set Up A User ID For Batch Jobs

When batch jobs execute, they must reference a given user's information for the following:

- Batch jobs can cause "auditable" events to occur. These events have a user ID associated with them.
- Some batch jobs access language-specific data, the language used is defined on a given user.
- When batch jobs complete, an email message is sent to a user's email address.

When a batch job is submitted, it references a specific user ID; this user ID is used for the above points. We recommend setting up a "dummy" user for this purpose (e.g., user ID = `BATCH`).

Set Up A Feature Configuration

After completing the above set up tasks, you must set up a *Feature Configuration* to provide the batch scheduler with the information it needs to operate.

The following points describe the various **Option Types** that must be defined for the feature configuration:

- `Active Job Stream (WF Template Code).` Set up a separate option for every job stream. The option's value is the Workflow Process Template associated with the job stream. Note, only these job streams appear on the *Job Stream Summary* page and are available in the *Job Stream Creation Schedule* page.

- `Batch Code FK Char Type.` Set up a single option to define the foreign key characteristic type used to reference the batch code on the workflow events. Refer to *Set Up Characteristic Types* for more information.
- `Batch Code of WF Event Activation Process.` Set up a single option to define the batch code of the batch job that is responsible for *activating the workflow events* that cause `Pending` batch jobs to be created.
- `Batch Job ID FK Char Type.` Set up a single option to define the foreign key characteristic type used to reference the batch job on the workflow events. Refer to *Set Up Characteristic Types* for more information.
- `Batch Run Number Ad hoc Char Type.` Set up a single option to define the ad hoc characteristic type used to reference the batch run's run number on the workflow events. Refer to *Set Up Characteristic Types* for more information.
- `Business Date Ad hoc Char Type.` Set up a single option to define the ad hoc characteristic type used to reference the batch run's business date on the workflow process (a single business date is used for all batch jobs initiated by a job stream). Refer to *Set Up Characteristic Types* for more information.
- `Email Address on Batch Jobs.` Set up a single option to define the email address used to notify a user when a batch job completes (note, the Email's subject line indicates if the batch job failed or succeeded). Note, you can override the email address on a specific batch job using a parameter on the Workflow Event Activation algorithm used to create the `Pending` batch job. If you leave this field blank (and no override is defined for the batch job), no email will be sent.
- `SMTP Server Name` and `SMTP Port Number`. If you want email sent informing of the completion of a batch job, define the SMTP server and port number.
- `User ID on Batch Jobs.` Set up a single option to define the user ID controlling user-oriented functionality. Refer to *Set Up A User ID For Batch Jobs* for more information.
- `Work Calendar.` Set up a single option to define the work calendar used when Workflow Process Recurrence Schedules are created. Refer to *Set Up A Work Calendar* for more information.

Maintaining Job Stream Creation Schedules

As described under *How To Start A Job Stream*, you can set up a job stream creation schedule to define when a job stream (workflow process) should be automatically created by the system. Open **Admin Menu, Job Stream Creation Schedule** to set up a job stream creation schedule.

Description of Page

On this page, you define when the system should automatically create a workflow process for a job stream template. To do this:

- Enter the **Job Stream Template**.

Note: Only job streams defined on the *Feature Configuration* for the Batch Scheduler appear on this page.

- Enter the **Dates and Times** on which the system should automatically create the workflow process.

When a schedule record is added, the system sets its status to `Pending`.

Click the **Hold** button to change a `Pending` schedule to `Hold` if you don't want the system to create a workflow process on the date and time. You'd hold a record rather than delete it if you intend to release the hold in the near future.

Click the **Release** button to change a `Held` schedule to `Pending` if you want to release the hold.

When the system creates a workflow process for a job stream schedule record, the schedule record's status is changed to `Process Created`. There is one exception to this statement - if multiple `Pending` schedule records exist for a given workflow process template, the system will create a single workflow process for the one with the latest date / time and mark all others as `Skipped Due To Overlap`.

Schedule records in the `Process Created` and `Skipped Due To Overlap` states are protected.

Rather than entering the individual Dates and Times, you can press the **Recurrence** button to have the system generate these for you. Pushing this button causes the **Recurrence Window - Daily Pattern** to open.

The **Range From** and **To** define the dates during which recurring entries will be created.

The **Pattern** defines the frequency of recurrence:

- The `Daily` pattern will create a schedule for every date in the range defined above (subject to exceptions defined below). You can define the following for this pattern:
  - Use **Repeat Every** to define if the schedule should be created every day (a value of `1`), every other day (a value of `2`), every third day (a value of `3`), etc.
  - Enter the **Start Time** defined on the schedule records.
  - Turn on **Include Weekends** if schedule records should also be created on weekends.
- The `Hourly` pattern will create a schedule for every hour in the date range defined above (subject to exceptions defined below). You can define the following for this pattern:
  - Use **Repeat Every** to define if schedule records should be created every hour (a value of `1`), every other hour (a value of `2`), every third hour (a value of `3`), etc.
  - Enter the **Between** start time and end time to restrict the hours during which schedule records are created.
  - Turn on **Include Weekends** if schedule records should also be created on weekends.
- The `Weekly` pattern will create a schedule for every week in the date range defined above (subject to exceptions defined below). You can define the following for this pattern:
  - Enter the **Start Time** defined on the schedule records.
  - Select the day(s) of the week that the schedule records should be created.

How To Copy Job Streams From The Demonstration Database

> **Note: Workflow Process Template = Job Stream.** Please note that a separate workflow process template exists for each job stream. When you copy job streams from the demonstration database, you are actually copying workflow process templates.

The demonstration database contains many sample workflow process templates. The topics in this section describe how to copy these templates to your implementation's database.

If You Work In A Non-English Language

The demonstration database is installed in English only. If you work in a non-English language, you must execute the *NEWLANG* background process on the demonstration database before using it as a `Compare Source` supporting environment. If you work in a supported language, you should apply the language package to the demonstration database as well.

If you don't execute `NEWLANG` on the demonstration database, any objects copied from the demonstration database will not have language rows for the language in which you work and therefore you won't be able to see the information in the target environment.

One Time Only - Set Up a DB Process To Copy Workflow Process Templates

You need to copy workflow process templates in the target database (i.e., your implementation's database) that looks like the following:

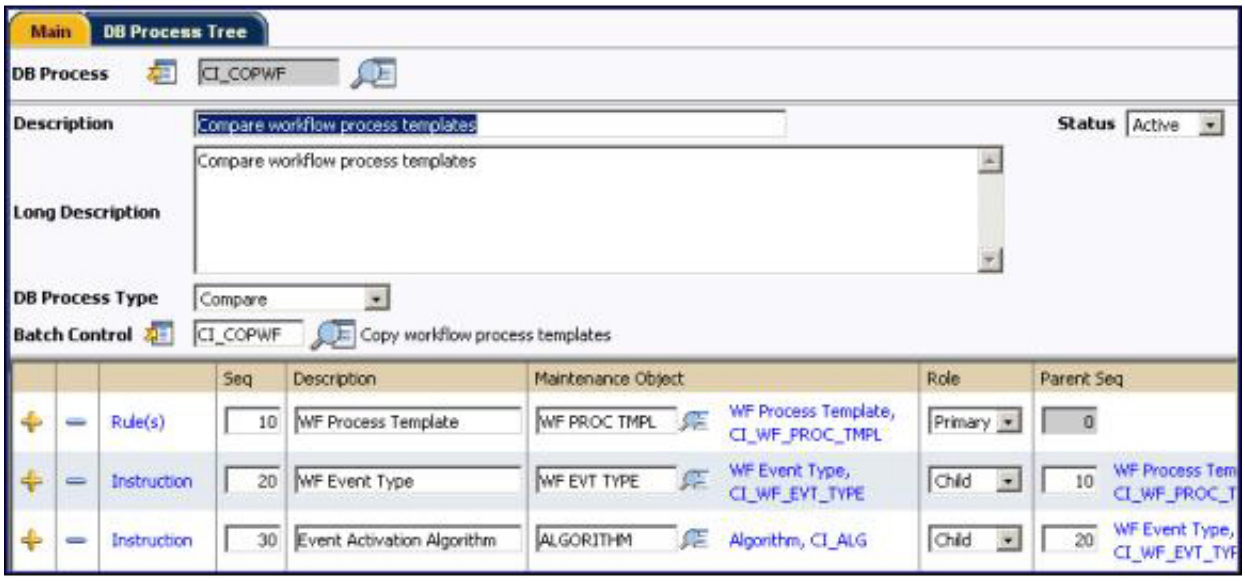


Figure 7: DB Process To Copy Workflow Process Templates

The demonstration database contains such a DB process; it's called `CI_COPWF`. In order to copy workflow process templates from the demonstration database, you must first copy this DB process from the demonstration database.

Caution: The remainder of this section is confusing as it describes a DB process that copies another DB process. Fortunately, you will only have to do the following once. This is because after you have a "copy workflow process templates" DB process in your target database, you can use it repeatedly to copy workflow process templates from the demonstration database.

You can copy the `CI_COPWF` DB process from the demonstration database by submitting the `CL-COPDB` *background* process in your target database. When you submit this process, you must supply it with an environment reference that points to the demonstration database. If you don't have an environment reference configured in your target database that references the demonstration database, you must have your technical staff execute a registration script that sets up this environment reference.

`CL-COPDB` is initially delivered ready to copy *every* DB process that is prefixed with `CI_` from the source database (there are numerous sample DB processes in the demonstration database and this process copies them all). If you only want to copy the `CI_COPWF` DB process, add a table rule to the primary instruction of the `CL-COPDB` *database process* to only copy the `CI_COPWF` DB process. The remainder of this section assumes you have added this table rule.

When the `CL-COPDB` process runs, it highlights differences between the "copy scripts" DB process in your source database and the target database. The first time you run this process, it creates a root object in the target database to indicate the `CI_COPWF` DB process will be added to your target database.

Note: Automatic approval. When you submit `CL-COPDB`, you can indicate that all root objects should be marked as `approved`.

After you've approved the root object(s), submit the `CL-APPCH` batch process to change your target database. You must supply the `CL-APPCH` process with two parameters:

- The DB Process used to create the root objects (`CL-COPDB`)
- The environment reference that identifies the source database (i.e., the demonstration database)

Run The Copy Workflow Process Templates DB Process

After you have a "copy workflow process templates" DB process in the target database, you should add a table rule to its primary instruction to define which workflow process template(s) to copy from the demonstration database. For example, if you want to copy a single workflow process template called `CI_DAILY`, you would have the following table rule:

- **Table:** CI_WF_PROC_TMPL (WF Process Template)
- **Override Condition:**#CI_WF_PROC_TMPL.WF_PROC_TMPL_CD = 'CI_DAILY'

If you do not introduce this table rule to the primary instruction of the DB process, ALL workflow process templates in the demonstration database will be copied to the target database (and this may be exactly what you want to do).

Note: Tables rules are WHERE clauses. A table rule is simply the contents of a WHERE clause except the tables names are prefixed with #. This means that you can have table rules that contain LIKE conditions, subqueries, etc.

At this point, you're ready to submit the background process identified on your "copy workflow process templates" DB process. This background process highlights the differences between the workflow process templates in the demonstration database and the target database (the target database is the environment in which you submit the background process).

Note: The background process you submit is typically named the same as the DB process that contains the rules. If you used the `CL-COPDB` background process to transfer the "copy workflow process templates" DB process from the demo database, it will have also configured a batch control and linked it to the "copy workflow process templates" DB process. This batch control has the same name as its related DB process (this is just a naming convention, it's not a rule). This means that you'd submit a batch control called `CI_COPWF` in order to execute the `CI_COPWF` DB process.

When you submit the `CI_COPWF` background process, you must supply it with an environment reference that points to the source database (i.e., the demonstration database).

When the `CI_COPWF` process runs, it simply highlights differences between the workflow process templates in your source database and the target database. It creates a root object in the target database for every workflow process template that is not the same in the two environments (actually, it only concerns itself with workflow process templates that match the criteria on the table rule described above).

Note: Auto approval. When you submit `CI_COPWF`, you can indicate that all root objects should be marked as `approved` (thus saving yourself the step of manually approving them).

After you've approved the root object(s) associated with the workflow process template(s) that you want copied, submit the `CL-APPCH` batch process to cause your target database to be changed. You must supply the `CL-APPCH` process with two parameters:

- The DB process of the "copy workflow process templates" DB process (i.e., `CI_COPWF`)
- The environment reference that identifies the source database (i.e., the demonstration database)

Chapter 18

Configuring Zones

Topics:

Many zones in Oracle Revenue Management and Billing do not require configuration by your implementation team. This zone does not require configuration because its zone type has no configurable options (i.e., its behavior is static).

Other zones require configuration before they can be used because their behavior is dynamic. The topics in this section provide tips and techniques on how to configure zones in Oracle Revenue Management and Billing.

Fastpath: Refer to *The Big Picture of Portals and Zones* for a description of portal and zone functionality.

Configuring Timeline Zones

A timeline zone contains one or more "lines" where each line shows when significant events have occurred. For example, you can set up a timeline zone that has two lines: one that shows when payments have been received from a customer, and another that shows when bills have been sent to the customer.

The following points describe how to set up a timeline zone:

- Set up an *algorithm* for each line in the zone. These algorithms will reference an algorithm type that is plugged into the `Zone - Timeline Line` plug-in spot. Click *here* to see the algorithm types available for this plug-in spot. Please note the following about the parameter values defined on these algorithms:
    - You can set up a timeline algorithm to show an object's "info string" when a user clicks on an event on a timeline. The object's info string appears in the zone's info area. When a user clicks on an "info string", they are transferred to a page (typically the one used to maintain the object). For example, if a user clicks on a "bill info" line, they will be transferred to the bill maintenance page.

You control the format of the info string and the destination transaction by defining the appropriate *foreign key reference* in each timeline algorithm's parameters. For example, if you were setting up the algorithm for a bill line, you'd reference the foreign key reference used to show bill foreign keys throughout the system.

- You can set up a timeline algorithm to show *BPA scripts* when a user clicks on an event on a timeline. For example, if you click on a bill event, BPA script descriptions can appear in the info area. When a user clicks on one of these descriptions, the script will execute and guide them through a respective business process (e.g., initiate a bill dispute, request a bill reprint, etc.). You define the scripts in each timeline algorithm's parameters.

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 bill event is kicked off, the system populates a field called BILL_ID with the prime-key of the selected bill.

The script can use these page data model field to navigate to the pertinent pages. For example, if you were setting up a script to reprint a bill, the first line of the script would reference a navigation option to transfer the user to the Bill - Routing page where they can initiate the reprint. This navigation option will contain context fields that matched the names of the fields in the page data model (this is how field values are passed to pages).

- You can control every color and icon shown on a timeline by specifying the appropriate color codes on the zone's parameters.
- Set up a *zone* that references these algorithms. The zone will reference the `F1-TIMELINE` zone type.
- Update your users' portal preferences and *security rights* so they can see the zone in the desired location on the portal(s).

You can set up many timeline zones. For example,

- You might want different zones to appear on a portal depending on the type of user. For example, you might want one timeline for billing clerks, and a different one for customer service representatives.
- For aesthetic reasons, you might want multiple simple timeline zones to appear on a given portal rather than one complex timeline zone.
- You might want to set up context specific timeline zones. For example, you might want to have one timeline zone that is person-oriented.

Chapter 19

CTI-IVR Integration

Topics:

Oracle Revenue Management and Billing provides tools to facilitate the integration with your Computer Telephony Integration/Interactive Voice Response (CTI/IVR) system. The interface provides the following functionality:

- The ability to launch Control Central for a particular account ID or phone number from an external application
- The ability to perform an outbound phone call from within Oracle Revenue Management and Billing
- The ability to accept the next call, as dictated by the CTI software, from the toolbar

This document provides technical information needed by your implementers to fully integrate with your CTI/IVR system.

Launching The System From an External Application

There following sections describe possible options to launch the system from an external system.

Launching Control Central Using an ActiveX Navigator

Oracle Revenue Management and Billing provides an ActiveX component `CDxCTI.CDxNavigator` that external applications, such an IVR application, can use to launch Control Central for a given account number or phone number.

Note: Enable ActiveX. In order to use the navigator, you must *configure your browser to enable ActiveX*.

The CDxNavigator object exposes two methods:

- ***ContralCentralByAccountId*** invokes Control Central for a given account ID.
- ***ControlCentralByPhone*** invokes Control Central for a given phone number.

Method: ControlCentralByAccount

Input:

- **Server URL:** The Oracle Revenue Management and Billing server URL, for example http://spl-server
- **AccountId:** The account ID to search for.

VB Example: Navigate to Control Central Using an Account ID

Dim nav As New CDxTAPI.CDxNavigator

nav.ControlCentralByAccountId ("http://spl-server:1000", AccountID)

Web Page Example: Navigate to Control Central Using an Account ID

Dim nav As New ActiveXObject("CDxTAPI.CDxNavigator");

nav.ControlCentralByAccountId (" *http://spl-server:1000*", AccountID)

Method: ControlCentralByPhone

Input:

- **Server URL:** The Oracle Revenue Management and Billing server URL, for example http://spl-server
- **PhoneNumber**: The phone number of the person to search for
- **PhoneFormat:** The format in which the phone number is provided

VB Example: Navigate to Control Central Using a Phone Number

Dim nav As New CDxTAPI.CDxNavigator

nav.ControlCentralByPhone (" *http://spl-server:1000* ", "(415) 357-5423", "(999) 999-9999")

Web Page Example: Navigate to Control Central Using a Phone Number

Dim nav As New ActiveXObject("CDxTAPI.CDxNavigator");

nav.ControlCentralByPhone ("http://spl-server:1000", "(415) 357-5423", "(999) 999-9999");

Main Processing

The ActiveX component performs the following:

- Locate the first Internet Explorer instance running Oracle Revenue Management and Billing.

- If an Internet Explorer session is found, use ActiveX automation to navigate to Control Central. Depending on the search type (account or phone), enter the appropriate values in the Control Central page and launch the search.
- If an Internet Explorer session cannot be found, launch Internet Explorer and go directly to Control Central.

Navigation Sample Provided with the System

An HTML page has been provided to demonstrate the integration. This sample may be found in the following location on your Oracle Revenue Management and Billing server: /ci/cti/CTISample.HTM.

- Start an Internet Explorer session.
- Navigate to URL above.
- Select an account ID or phone number and click **Navigate**.

Note: Sample Data. The accounts and phone numbers included in the sample HTML file correspond to accounts and phone numbers that exist in the demo database.

- A new session is started if one is not already active and Control Central is launched with the account corresponding to the selected account or phone number.

This sample program is provided to illustrate to implementers how to integrate their CTI-IVR system with Oracle Revenue Management and Billing.

Launching The Application Using a URL

You may also launch the application using a URL. With this option you can set the system to launch a script upon startup. You can also indicate to the system to automatically load an appropriate page (if this information is not part of the script).

Fastpath: Refer to *Launching A Script When Starting The System* for further information.

Initiating an External Call

This section describes the automated dialer functionality provided with the system as well as information about integrating with your own automated dialer.

Overview of Automated Dialer

In order to initiate a call to a customer from within the system, a context menu item ***Go To Automated Dialer*** is available on the Person context menu. To call a customer displayed in the current context, choose this option from the person context menu and a window appears, showing a list of phone numbers defined for that person.

Select the desired phone number and click **Dial**.

Note: Context Entry Secured. The *navigation key* for this window `automatedDialer` refers to an application service to facilitate application security. If your installation does not support an integration with external dialer software, configure the security settings to ensure that users do not have access to the application service for this context entry.

Technical Implementation of Automated Dialer

The popup window is implemented as a JSP page, which calls the JSP page ext_cti_dialer.jsp to integrate with an automated dialer. The ext_cti_dialer.jsp page provided with the system integrates to the Microsoft Phone Dialer, available on any Windows 2000 workstation.

The Microsoft Phone Dialer is invoked through the CDxPhoneDialer ActiveX object.

****

* Invoke an external phone Dialer

* In the sample provided we launch the Microsoft phone Dialer

****

*/

function callDialer(phoneNumber){

CDxPhoneDialer.makeCall(phoneNumber);

}

Object: CDxPhoneDialer

This object is used to call the Microsoft Phone Dialer for an outbound call. It is only used when your implementation uses the Microsoft standard dialer.

Method: makeCall

Input: Phone Number

Microsoft Phone Dialer Configuration

If your implementation chooses to use the functionality provided with the system and integrate with Microsoft Phone Dialer, you must perform the following steps:

- Copy the JSP page ext_cti_dialer.jsp from the /cm_templates directory found under the web application root directory on your Oracle Revenue Management and Billing server to the /cm directory.

Note: Enable ActiveX. In order to use the integration with the Microsoft Phone Dialer, you must *configure your browser to enable ActiveX*.

Customize Integration to Your Automated Dialer Software

In order to integrate with a different automated dialer software application, your implementers must modify the ext_cti_dialer.jsp to call the appropriate dialer.

- Copy the JSP page ext_cti_dialer.jsp from the /cm_templates directory found under the web application root directory on your Oracle Revenue Management and Billing server to the /cm directory.
- Make the appropriate changes to the copy of ext_cti_dialer.jsp in the /cm directory to integrate with your automated dialer.

Customize Automated Dialer User Interface

Your implementation may choose to display a different user interface for the **Go To Automated Dialer** function than the one provided with the system. For example, perhaps there is more information that you would like to display in addition to the person's name and phone numbers. In order to do this, perform the following steps:

- Create your customized component to provide the desired functionality.
- Create a navigation key for your new component and indicate the URL being overridden. The remainder of the section walks you through these steps.

Go to **Admin Menu, Navigation Key** +.

For **Navigation Key,** specify a name for the new navigation key prefixed with CM.

For **URL Location**, select `External (Override)` to override a base navigation key.

When you select `External (Override)`, the **Overridden Navigation Key** becomes available. Select the `automatedDialer` navigation key because that is the key you are overriding.

The **URL Override** is the path on the Web server to your custom component.

When overriding a navigation key, you must flush the system login cache on the Web server. The navigation keys are stored in the system login cache, so the overrides do not become effective until the cache is flushed. To flush the cache, issue the following command in your browser's address bar: `http://server:port/flushSystemLoginInfo.jsp`, where server is the name or address of your web server and port is the port number of the application, for example, `http://CD-Implementation:7500/flushSystemLoginInfo.jsp`.

Fastpath: Refer to the *Defining Navigation Keys* for more information.

Receiving the Next Caller in the Queue

If your CTI-IVR system allows users to request the next caller waiting in a queue, the system provides a mechanism to integrate with this functionality.

A ***Next Call*** button is available in the toolbar that can be used to request the next call waiting in an inbound queue managed by a CTI application. This button is only enabled if you have set the **CTI Integration** flag to `Yes` on your *installation options*.

When the next call button is clicked, it launches a browser script function called `launchCTI` located in a file called `ext_cti.jsp`. The `launchCTI` function calls a function called `ctiGetNextCaller` to retrieve the next caller's account ID and uses the `CDxNavigator` object to launch Control Central for that account.

Customize Integration to Your Next Caller Function

The ext_cti.jsp file shipped with the base product provides sample functionality that should be replaced with the appropriate integration to your CTI application. In the sample provided, the `ctiGetNextCaller` randomly takes an account ID from a predefined list of accounts.

In order to integrate the next caller functionality with your CTI-IVR system, perform the following steps:

- Copy the JSP page ext_cti.jsp from the /cm_templates directory found under the web application root directory on your Oracle Revenue Management and Billing server to the /cm directory.
- In the /cm directory, replace the contents of the `ctiGetNextCaller` function to retrieve the next caller ID from your CTI application.

ActiveX Component - CDxCTI

The system provides an ActiveX component ***CDxCTI*** that contains all the functionality required for inbound and outbound calling. It contains two objects:

- CDxNavigator
- CdxPhoneDialer

Configuring Your Browser to Enable ActiveX

In order to use the automated phone dialer functionality or the next caller functionality, your users must configure their browser to enable ActiveX. Perform the following steps:

- In your Internet Explorer browser window, navigate to **Tools**, **Internet Options** and go to the **Security** tab. From there, select **Local Intranet**.
- Click **Custom Level**.
- Under the ActiveX controls and plug-ins section, set the following:
  - Download signed ActiveX controls: `Prompt`
  - Download unsigned ActiveX controls: `Disable`

- Initialize and script ActiveX controls not marked as safe: `Disable`
- Run ActiveX controls and plug-ins: `Enable`

Installing the CDxCTI ActiveX Component

The CDxCTI ActiveX components install automatically the first time the Automated Phone Dialer is launched or when the CTISample.htm is launched. The CDxCTI component is signed using Microsoft Authenticode technology, therefore when it is downloaded the first time, a dialog will appear describing the source of the component and asking the user to accept the installation of the component on the local machine.

Creating an Instance of the CDxCTI Object in a Web Page

To use the CDxCTI objects from a web page, declare them explicitly using the OBJECT tag:

<OBJECT ID="CDxPhoneDialer"

CLASSID="CLSID:151A6E91-8C55-4666-BFFB-9EC345583CBD"

CODEBASE="CDxCTI.CAB#version=1,5,0,8">

</OBJECT>

<OBJECT ID="CDxNavigator"

CLASSID="CLSID:E7EF882D-662A-4451-A78C-CD62393F06C6"

CODEBASE="CDxCTI.CAB#version=1,5,0,8">

</OBJECT>

Alternatively, use the new ActiveXObject function

var nav = new ActiveXObject("CDxCTI.CDxNavigator");

var nav = new ActiveXObject("CDxCTI.PhoneDialer");