Oracle Enterprise Taxation Management

Administration Guide

Release 2.3.0

E21669-01

January 2011

ORACLE

Oracle Enterprise Taxation Management Administration Guide

E21669-01

Table of Contents

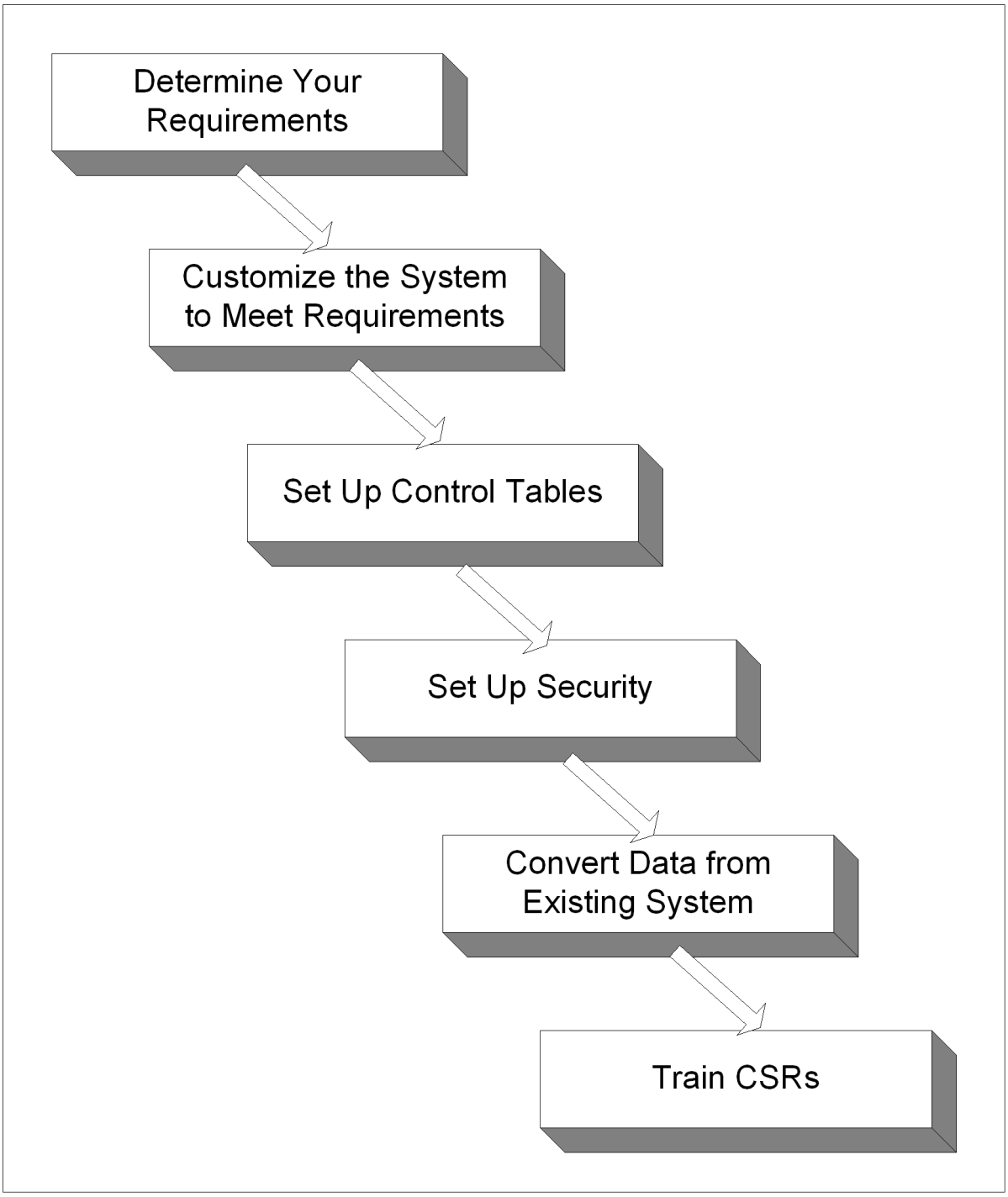
Oracle Enterprise Taxation Management Administration

The topics in this section describe how to administer the Oracle Enterprise Taxation Management application.

Preparing To Implement

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 Enterprise Taxation Management, you may also have discovered that the product 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 record*.	You can run the **CI_COPIN** DB process to copy many of the algorithms that support basic functionality from the demonstration database. Refer to *How To Copy An Algorithm From The Demo Database* for more information.
Accounting Environment		
Country & State	Admin Menu, Country	
Currency Codes	Admin Menu, Currency Code	**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	
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 set up. It references all other application services and a single user called **SYSUSER**. Note: You may be able to *import sample user groups from the demonstration database*. Also, 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 set up: **NORTHAM** displays currencies and dates in a classic American format; **EURO**

Function	*Menu*	*Auto Setup*
		displays information in a classic European format
Data Access Role	Admin Menu, Data Access Role	**CI_DFLT** is automatically populated. It references a single user called **SYSUSER** and a single access group called **CI_ACCGRP**
Access Group	Admin Menu, Access Group	**CI_ACCGRP** is automatically populated. It references the data access role **CI_DFLT**
User	Admin Menu, User	**SYSUSER** is automatically set up. Note, you may be able to *import your users 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	
Account Type Environment		
Account Type	Admin Menu, Account Type. At this point, you'll only be able to set up your account type codes. You will return to these account types throughout the setup process to populate additional information.	
Financial Transaction Environment		
Work Calendar	Admin Menu, Work Calendar	
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 rate quantities onto billable charges created using this template, you must set up the appropriate unit of measure code, time-of-use code and/or rate quantity identifier.	
Billable Charge Upload 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	Rather than setting these up manually, you can run the **CI_COPBI** DB process to copy many of these algorithms from the demonstration database. Please review

Function	Menu	Auto Setup
	bill lines, 3) construct a bill segment's financial transaction, 4) cancel previously estimated bill segments	the parameter values on these algorithms after they are copied. Refer to *How To Copy An Algorithm From The Demo Database* for more information.
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	Rather than setting these up manually, you can run the **CI_COPPY** DB process to copy many of these algorithms from the demonstration database. Please review the parameter values on these algorithms after they are copied. Refer to *How To Copy An Algorithm From The Demo Database* for more information.
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	Rather than setting these up manually, you can run the **CI_COPAD** DB process to copy many of these algorithms from the demonstration database. Please review the parameter values on these algorithms after they are copied. Refer to *How To Copy An Algorithm From The Demo Database* for more information.
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.	
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 record*.	
Debt Category	Admin Menu, Debt Category	
Debt Category Priority	Admin Menu, Debt Category Priority	
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	

Function	*Menu*	*Auto Setup*
	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	
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 and the *Messages* tab in the Framework page for more information.	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm that distributes payments.	If you ran the **CI_COPPY** DB process described above, this algorithm will have been set up for you.
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 taxpayers can have individual bill due dates.	
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 record*.	You can run the **CI_COPIN** DB process to copy many of the algorithms that format basic information from the demonstration database. Refer to *How To Copy An Algorithm From The Demo Database* for more information.
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 record*.	If you ran the **CI_COPIN** DB process described above, this algorithm will have been set up for you
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	If you ran the **CI_COPIN** DB process described above, this algorithm will have been set up for you

Function	Menu	Auto Setup
	amount of an automatic payment for a bill for an account with an active auto pay option. This algorithm is plugged-in on the *installation record*.	
Algorithm	Admin Menu, Algorithm. Refer to *Account Type* for other available plug-in spots that may be used by your implementation to perform additional logic when processing payments and bills.	
Return to Account Type	Admin Menu, Account Type. You will need to plug-in the algorithms defined above on your account types.	
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 set up account management groups. Refer to *Assigning A To Do Role* for more information on how account management groups may be used to define an entry's role.	
Account Relationship	Admin Menu, Account Relationship Type	
Alert Type	Admin Menu, Alert Type	
Bill Message	Admin Menu, Bill Message	
Algorithm	Admin Menu, Algorithm. If you have software 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.	Refer to *How To Copy An Algorithm From The Demo Database* for more information.
Bill Route Type	Admin Menu, Bill Route Type	
Contract Quantity Type	Admin Menu, Contract Quantity Type	
Algorithm	Admin Menu, Algorithm. If you have software 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.	If you ran the **CI_COPD1** DB process described above, these algorithms will have been set up for you.
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*
Identifier Type	Admin Menu, Identifier Type	
Industry Codes	Admin Menu, Industry 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.	
Person Type	Admin Menu, Person 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 record*.	If you ran the **CI_COPIN** DB process described above, this algorithm will have been set up for you
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 record*.	If you ran the **CI_COPIN** DB process described above, this algorithm will have been set up for you
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 record*.	
Algorithm	Admin Menu, Algorithm. If you have software capable of reconstructing an image of a letter in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the *installation record*.	If you ran the **CI_COPD1** DB process described above, this algorithm will have been set up for you
Algorithm	Admin Menu, Algorithm. If you have software capable of reconstructing an image of a bill in a PDF for the purpose of online display, you will need to create an algorithm that renders this PDF. This algorithm is plugged-in on the *installation record*.	If you ran the **CI_COPD1** DB process described above, this algorithm will have been set up for you
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.	
Automatic Payment (EFT) Environment		

Function	***Menu***	***Auto Setup***
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm to create automatic payments. This algorithm is plugged-in on the *installation record*.	You can run the **CI_COPAP** DB process to copy this algorithm (and other autopay-oriented algorithms) from the demonstration database. Refer to *How To Copy An Algorithm From The Demo Database* for more information.
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.	If you ran the **CI_COPAP** DB process described above, this algorithm will have been set up for you
Auto Pay Route Type	Admin Menu, Auto Pay Route Type	
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 taxpayer's bank account or credit card number, you will need to set up the appropriate validation algorithms.	If you ran the **CI_COPAP** DB process described above, this algorithm will have been set up for you
Auto Pay Source Type	Admin Menu, Auto Pay Source Type	
Algorithm	Admin Menu, Algorithm. You may need to set up an algorithm if your taxpayers can define a maximum withdrawal limit on their autopay options.	If you ran the **CI_COPAP** DB process described above, this algorithm will have been set up for you
Return to Account Type	Admin Menu, Account Type. You should plug-in the Autopay Over Limit Algorithm in each appropriate Account Type.	
Characteristics		
Algorithm	Admin Menu, Algorithm. If you have ad hoc characteristic types, you may need to	

Function	*Menu*	*Auto Setup*
	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	
Location Environment		
Location Type	Admin Menu, Location Type	
Algorithm	Admin Menu, Algorithm. You will need to set up an algorithm to format the standard location info that appears throughout the system. This algorithm is plugged-in on the *installation record*.	If you ran the **CI_COPIN** DB process described above, this algorithm will have been set up for you
Algorithm	Admin Menu, Algorithm. You may need to set up the algorithms that determine if geographic ID's are in a predefined format.	
Geographic Type	Admin Menu, Geographic Type	
Rate Environment		
Frequency	Admin Menu, Frequency	
Rate Quantity Identifier	Admin Menu, Rate Quantity Identifier	
Algorithm Type	Admin Menu, Algorithm Type. If you create new Rate 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 RQ rule).	All base package algorithm types are automatically populated
Rate Quantity Rule	Admin Menu, Rate Quantity Rule	
Rate Factor	Main Menu, Rates, Rate Factor	
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	
Rate Factor Value	Main Menu, Rates, Rate Factor Values	

Function	*Menu*	*Auto Setup*
Obligation Configuration		
Filing Calendar	Admin Menu, Filing Calendar	
Tax Type	Admin Menu, Tax Type	
Algorithm	Admin Menu, Algorithm. You will need to set up the algorithms that determine: • Special criteria to be tested before an obligation is severed. • Special processing that should take place prior to the completion of a bill that references obligations of a given type. • Special processing that should take place during completion of a bill that references obligations of a given type. • Special processing that should take place when obligations of a given type are created. • Special processing that should take place when a financial transaction is frozen for obligations of a given type.	
Algorithm	Admin Menu, Algorithm. You may want to set up an algorithm that formats the obligation information that is displayed throughout the system. This algorithm is plugged-in on the *installation record*.	
Algorithm	Admin Menu, Algorithm. You may want to set up an algorithm that formats the obligation information that is displayed throughout the system for a specific obligation type. This algorithm is plugged-in on the *Obligation Type*.	
Bill Cycle, Bill Cycle Schedule	Admin Menu, Bill Cycle	
Bill Period, Bill Period Schedule	Admin Menu, Bill Period	
Obligation Type	Admin Menu, Obligation Type	
Obligation Type Start Options	Admin Menu, Obligation Type Start Option	
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)	
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.	

Function	*Menu*	*Auto Setup*
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 obligation. Refer *How To Get An Unbalanced Tender Control In Balance (Fixing Over/Under)* for more information.
- Create an obligation to which your over/under payments will be linked. This obligation will reference your over / under obligation type. Refer to *Over / Under Cash Drawer Obligations* for more information.

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

- Create a person and account to which the system will link payments with invalid account. Refer to *Phase 3 - Create Payment Events, Tenders, Payments and Payment Segments* for information about the process that books invalid payments to this account. Refer to *How To Transfer A Payment From One Account To Another* for how a user transfers the payment from the invalid account to the correct account (once known).
- Create an obligation for this account. This obligation will reference your payment suspense obligation type. The system needs this obligation 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 Obligations* for more information.
- Update the tender source associated with the respective source of payments to indicate the obligation created in the previous step should be used for payments with invalid accounts. Refer to *Setting Up Tender Sources* for more information.
- Because the payment upload process simply books payments that reference invalid accounts to the account associated with the suspense obligation on the payment's tender source, this account should belong to an account type with the appropriate payment distribution algorithms. This may entail creating a new account type that will only be used on suspense accounts.

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

Cross Reference To The Remaining Chapters

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

- Refer to *Defining General Options Addendum* and *Defining General Framework Options* for a discussion of the control tables associated with general functionality (e.g., country codes, state codes, etc.).
- Refer to *Defining Financial Transaction Options* for a discussion of the tables affecting your financial transactions (e.g., bill segment types, payment segment types, etc.).
- Refer to *Defining Taxpayer Options* for a discussion of the control tables affecting persons, accounts and obligations.
- Refer to *Rates* for a discussion of the control tables affecting your rates.
- Refer to *Defining Obligation Types* for a discussion of the control tables affecting your obligation types.
- Forms processing tables and form rules need to be set up for processing registration forms, tax forms and form upload staging tables. Refer to *Defining Forms Processing Options* and *Defining Form Rules* for more information.
- Penalty and interest related configuration tables need to be set up for penalty and interest calculations. Refer to *Setting Up Penalty and Interest Options* for a description of tables that must be set up to enable this functionality.
- Overpayment processing tables need to be set up for overpayment activity. Refer to *Setting Up Overpayment Options* for a description of tables that must be set up to enable this functionality.
- Overdue processing tables need to be set up for collection activity. Refer to *Overdue Processing - Setup Tasks* for a description of tables that must be set up to enable this functionality.
- Refer to *Setting Up Bankruptcy Options* for a list of objects to be defined for bankruptcy handling.
- Refer to *Setting Up Audit Case Options* for a list of objects to be defined for audit cases.
- Refer to *Setting Up Appeal Options* for a list of objects to be defined for appeals.
- Refer to *Setting Up Review Options* for a list of objects to be defined for reviews.
- Refer to *Setting Up Suppression Options* for a list of objects to be defined for suppression handling.

- Refer to *Defining Background Process* for a discussion of the control tables affecting your background processes.
- Refer to *Defining Algorithms* for a discussion of the control tables affecting the algorithms referenced on many control tables.

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.

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.

Reports Setup Sequence

In order to use the reporting tool, you will need to set up reporting options. Refer to *Configuring The System To Enable Reports* for more information.

Note:

Importing sample reports. Refer to *How To Copy A Report From The Demonstration Database* if you want to import sample report metadata from the demonstration database.

XML Application Integration Setup Sequence

In order to use the XAI tool for sending information between third parties, you will need to set up XAI control tables. Refer to *XML Application Integration* for more information.

Work Management Setup Sequence

Work management options need only be set up if your organization uses process flows to manage issues. Refer to *Setting Up Process Flows* for more information.

Zone Setup

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

To Do Options Setup

Refer to *Setting Up To Do Options* for more information on how to configure the system to match your organization's To Do management needs.

How To Copy An Algorithm From The Demonstration Database

Caution:

If you are not familiar with the concepts described in the *ConfigLab* chapter, this section will be difficult to understand. Specifically, you need to understand how a *Compare* database process is used to copy objects between two databases. Please take the time to familiarize yourself with this concept before attempting to copy an algorithm from the demonstration database.

The demonstration database contains many sample algorithms. The topics in this section describe how to copy a subset of the demonstration algorithms 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 Algorithms

You need a "copy algorithm" *database process* (DB process) set up in the target database (i.e., your implementation's database). This DB process has a single instruction that references the algorithm *maintenance object* (MO). This instruction should have a table rule with an override condition that selects the algorithms in question. For example, the override condition, #CI_ALG.ALG_CD IN ('ADJT-AC', 'ADJT-AD', 'ADJT-CA', 'ADJT-GL', ADJT-NM', 'ADJT-RA', 'ADJT-TC'), is used on the DB process that copies specific adjustment type algorithms.

The demonstration database contains several such a DB processes, for example:

- **CI_COPAD** copies algorithms that control how adjustment financial transactions are created.
- **CI_COPAP** copies algorithms that control how automatic payments are created.
- **CI_COPBI** copies algorithms that control how bill segments and their financial transactions are created.
- **CI_COPIN** copies various installation algorithms that control validation, formatting and other general functionality.
- **CI_COPPY** copies algorithms that control payment processing.
- …

In order to copy algorithms from the demonstration database, you must first copy these DB processes from the demonstration database.

⚠ **Caution:**

The remainder of this section is confusing as it describes a DB process that copies other DB processes. Fortunately, you will only have to do the following once. This is because you only have to copy these DB processes once.

You can copy these DB processes 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 set up 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. Refer to *Registering ConfigLab Environments* for more information.

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 only the DB processes described above, add a table rule to the primary instruction of the **CL-COPDB**database process to only copy the desired processes. If you don't add this rule, all DB processes in the demonstration will be copied to your target database (and this might be exactly what you want).

When the **CL-COPDB** process runs, it highlights differences between the DB processes in your source database and the target database. The first time you run this process, it creates a root object in the target database for each DB process that will be added to your target database. You can use the *Difference Query* to review these root objects and **approve** or **reject** them.

Note:

Automatic approval. When you submit **CL-COPDB**, you can indicate that all root objects should be marked as **approved** (thus saving yourself the step of manually approving them using *Difference Query*).

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 Algorithms DB Process

After you have populated the "copy algorithms" DB processes in your target database, you can override their *table rules* to edit the list of algorithms that will be copied. You need only do this if you don't need all of the algorithms that are defined in these DB processes (but it never hurts to have too many algorithms as they won't be used unless you plug them in on the appropriate object).

At this point, you're ready to submit the background process identified on your "copy algorithm" DB processes. These background processes highlight the differences between the algorithms 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 algorithm" DB processes from the demo database, it will have also set up these batch controls and linked to each the appropriate "copy algorithms" DB process. These batch controls have the same name as their related DB process (this is just a naming convention, it's not a rule). This means, for example, that you'd submit a batch control called **CI_COPAD** in order to execute the **CI_COPAD** DB process.

When you submit one of the DB processes defined above, you must supply it with an *environment reference* that points to the source database (i.e., the demonstration database).

When the process runs, it simply highlights differences between the algorithms in the source database and the target database. It creates a root object in the target database for every algorithm that is not the same in the two environments (actually, it only concerns itself with algorithm that match the criteria on the DB process's table rule described above). You can use the *Difference Query* to review these root objects and **approve** or **reject** them.

Note:

Auto approval. When you submit the process, 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 algorithms 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 algorithms" DB process (e.g., **CI_COPAD**)
- The environment reference that identifies the source database (i.e., the demonstration database)

Defining General Options Addendum

This section describes control tables that are used throughout Oracle Enterprise Taxation Management.

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 Enterprise Taxation Management 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*.

Use the **Location Geo Type** to indicate whether at least one geographic identifier (e.g., GPS coordinate) is **Required** or **Optional** on a location. Refer to *Defining Geographic Types* for more information.

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

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

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

Note:

Alerts that should appear adjacent to a person's name or address. If your organization doesn't use multiple character sets, you might want to consider using this functionality to implement critical person or location alerts. For example, if you have a taxpayer who's supported by a specific account representative, you could enter the account representative's name as the person's alternate name. If you do this, the account representative's name would appear in parenthesis following the taxpayer's name. In addition, you can search for the taxpayers supported by the account representative on Control Central by entering the account representative's name.

If your organization uses alternate representations of person name or address, set this flag to one of the following values:

- Use a value of **Address** if you only use alternate representations for location addresses.

- Use a value of **Name** if you only use alternate representations for a person's primary names.
- Use a value of **Name & Address** if you use alternate representations for both location addresses and person names.

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

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

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's* person/business identifier), the *Person* page defaults the identifier type defined in **Identifier Type (Person)**. For persons that are businesses (as defined by the *person type's* person/business identifier), the Person page 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 on the *Account* page, the system requires it have an account type. If the main taxpayer linked to the account is a human (as defined by the taxpayer's person type), the system defaults the account type defined in **Account Type (Person)**. For persons that are businesses (as defined by the person type), the system defaults the account type defined in **Account Type (Business)**. For more information, refer to *Setting Up Account Types*.

In addition to requiring an account type when a new taxpayer is added, the system also requires a "main taxpayer" (i.e., a reference to a person who is identified as the main taxpayer for the account). Enter the default **Account Relationship Type** code to be used to define the main taxpayer 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 taxpayer. For more information, refer to *Setting Up Bill Route Types*.

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 an obligation's balance and the general ledger are affected by bill segments and certain types of adjustments. Refer to *Preventing Obligation 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.

Installation Options - Collections

Select **Admin Menu** > **Installation Options** and use the **Collections** tab to define collections-specific installation options.

Description of Page

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

Use **Compliance Rating Threshold** to define when an account's compliance rating becomes risky. When an account's compliance rating falls beneath the Compliance Rating Threshold, the system will show an alert in Control Central highlighting such.

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. By default, the installation indicates that it is **Practiced**.

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" may be constructed using an algorithm plugged in here. Refer to *Defining Account Information* for more information on when this algorithm is used. Click *here* to see the algorithm types available for this system event.
Address Information	Optional	This algorithm allows your implementation to define a common format for displaying an address. It receives address constituents and returns a formatted address string. This plug-in spot may be invoked by other algorithms using the business service **C1-FormatAddressInfo**. 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" may be constructed using an algorithm plugged in here. Refer to *Adjustment Information* for more information on when this algorithm is used. Click *here* to see the algorithm types available for this system event.
Automatic Payment Creation	Required if you allow taxpayers 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.
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" may be constructed using an algorithm plugged in here. 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.
Compliance Rating "Created By" Information	Required	The data that appears in the compliance rating "created by" information is constructed using this algorithm. Refer to *Account - Collections* for more information about the compliance rating. Click *here* to see the algorithm types available for this system event.
Compliance Rating History Information	Optional	We use the term Compliance Rating History information to describe the basic information that appears throughout the system to describe a *compliance rating history* entry. Plug an algorithm into this spot to override the system default "compliance rating history information". Click *here* to see the algorithm types available for this system event.
Control Central Alert	Optional	There are two types of alerts that appear in the *Alert Zone* and on *Payment Event - Main*: 1) hard-coded system alerts and 2) alerts constructed by plug-in algorithms. You cannot change the hard-coded alerts (see the *Alert Zone* for the complete list). However, by plugging in this type of algorithm you can introduce additional alerts.

		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.
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 Obligation on a bill. Click *here* to see the algorithm types available for this system event.
Financial Transaction Info	Optional	We use the term "financial transaction information" to describe the basic information that appears throughout the system to describe a financial transaction. The data that appears in "financial transaction info" is constructed using this algorithm. Click *here* to see the algorithm types available for this system event.
Location Information	Required	We use the term "location info" to describe the basic information that appears throughout the system to describe a location. The data that appears in "location info" may be constructed using an algorithm plugged in here. Refer to *Defining Location Information* for more information on when this algorithm is used. Click *here* to see the algorithm types available for this system event.
Obligation Information	Optional	We use the term "obligation information" to describe the basic information that appears throughout the system to describe an obligation. The data that appears in "obligation information" may be constructed using an algorithm plugged in here. Refer to *Defining Obligation Information* for more information on when this algorithm is used. 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.
Override Proration Factors	Optional	This algorithm is only used if your organization has unusual rate proration requirements that necessitate the overriding of the base package proration logic. For example, you may have certain rate components whose charges should never be prorated. Refer to *Overriding Proration Factors* for more information. Click *here* to see the algorithm types available for this system event.
Override Seasonal Proration	Optional	This algorithm is only used if your organization has unusual method of determining the seasons for your rate components. For example, you may determine the seasonal boundaries for a rate component based on the scheduled meter read date associated with the bill cycle. Refer to the description of the seasonal attributes for a *rate component* 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.
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	We use the term "person information" to describe the basic information that appears throughout the system to describe a person. The data that appears in "person information" may be constructed using an algorithm plugged in here.

		Refer to *Person Information* for more information on when this algorithm is used. Click *here* to see the algorithm types available for this system event.
Person Name Validation	Required	The format of names entered on *Person - Main* is validated using this algorithm. Click *here* to see the algorithm types available for this system event.

Defining Taxpayer Languages

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

Each taxpayer'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 taxpayer of the account. In addition, the language code is passed on to all taxpayer-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 taxpayer'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 Calendars

The accounting calendar determines 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 obligation
- Every obligation references an obligation type
- Every obligation 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.

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 taxes reported on August 31. 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 division and a GL division. This is a rather powerful structure, but it can be confusing.

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

Note:

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

- A division is typically associated with a jurisdiction. The definition of a jurisdiction is a geographic-oriented entity with unique business rules. 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 choose **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*.

Defining Banks & Bank Accounts

The topics in this section describe how to maintain your implementation's bank accounts.

Bank - Main

To add or review Banks choose **Admin Menu** > **Bank**.

Description of Page

Enter a unique **Bank Code** and **Description** for the bank.

The **Bank Accounts** collection displays the bank accounts currently linked to this bank code. Use the drill down button to view more details or to modify the bank account details. Alternatively, you may navigate to the Bank Account tab and scroll to the desired bank account.

Bank - Bank Account

To add or review Bank Accounts for a Bank, choose **Admin Menu, Bank** and then navigate to the **Bank Account** tab.

Description of Page

Use the **Bank Accounts** tab to define the attributes of each bank account. For each account, enter the following information:

- Enter a **Bank Account Key** to identify an Account at a Bank. You may have more than one account at a given bank, and you may have accounts at more than one bank. This code will allow the system to easily identify a specific account.
- Enter a **Description** to appear on prompt lists, inquiries, and reports.
- Enter the **Account Number**, **Check Digit** and if needed, the **Branch ID** of the bank where the account is held.
- Enter the **Currency Code** for the currency in which the account is denominated.
- Use **DFI ID** to define the Depository Financial Institution ID that is interfaced to the automatic payment-processing agent as part of the automatic payment interface.
- Enter the **Distribution Code** to be used for cash GL distributions when a payment is frozen or canceled.

Where Used

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

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 Enterprise Taxation Management.

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 Enterprise Taxation Management may attempt to assign a role based on an account management group or division if it is applicable to the type of data related to the To Do entry.

As described in *The Big Picture of To Do Lists*, users are informed that something requires their attention by entries that appear in a To Do List. For example, consider what happens when billing can't find appraisal information for a property:

- The billing process creates a bill segment that is in error (appraisal information cannot be found).
- This bill segment that's in error, in turn, triggers the creation of a To Do entry.
- The To Do entry is assigned a role. A role is one or more users who can look at / work on the entry.
- When users view a To Do List, they only see entries addressed to roles to which they belong.

You can optionally use account management groups (AMG) to define the respective role to be assigned to the 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 the To Do entries that are associated with an account and To Do type. 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 the 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.

Configuring Zones

Many zones in Oracle Enterprise Taxation Management do not require configuration by your implementation team. For example, the base package is shipped with the **Account Financial History** zone that appears on the **Control Central - Account Information** portal. This zone does not require configuration because its zone type has no configurable options (i.e., its behavior is static).

Other zones require configuration before they can be used because their behavior is dynamic. The topics in this section provide tips and techniques on how to configure zones in Oracle Enterprise Taxation Management.

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 taxpayer, and another that shows when bills have been sent to the taxpayer.

Fastpath:

For a complete description of the numerous features available on a timeline zone, refer to *Timeline Zone - Account Info*.

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.
- Link the zone to the appropriate portal(s) (e.g., *Control Central - Account Information* or *Control Central - Taxpayer Information*).
- Update your users' *portal preferences* and *security rights* so they can see the zone in the desired location on the portal(s).

You can set up many timeline zones. For example,

- You might want different zones to appear on a portal depending on the type of user. For example, you might want one timeline for collection 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 location-oriented and another that is person-oriented.

Defining Financial Transaction Options

Bills, payments and adjustments share one very important trait - they affect how much your taxpayers 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.

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 taxpayer 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*.

Assessments Overview

An assessment is a tax liability financial transaction associated with an obligation and represents the legal presentation of a tax liability to a taxpayer. There are different types of assessments, based on how and why the liability was created, and the assessment type can apply differentiated processing rules, such as different penalty and interest calculation rules or different collections rules.

Most filing periods only have one assessment created that captures any applicable penalty, interest, or fees. However, it is possible for additional assessments to be created within a filing period.

Return and bill based tax types differ in how assessments get created and processed. The following sections discuss these concepts.

Return Based Taxes

Return based taxes are either expected or event based. Return based taxes represent taxpayers that have an expected obligation to file a Return for a filing period. Return based taxes can also be based on events such as taxpayers filing an excise tax. Income tax, withholding tax, and sales tax are examples of return based taxes.

For expected return-based tax types, the obligation is associated with a filing period as it tracks whether a filing obligation has been met for that period, and groups the assessments and financial transactions related to it. For most tax types, a filing period may be associated with a single type of obligation. For example, an active monthly Sales Tax filer has a single obligation for each month of the year. However, some tax types may require multiple filing obligations (of different types) over the same period of time. For example, a Withholding Tax type may have monthly Withholding Tax Returns as one obligation type, plus an annual Reconciliation Return and an Annual Taxpayer Withholding Detail statement as other obligation types.

Most return-based tax liability is created through the original self-assessment reported by a taxpayer on their tax return. There are two common models for self-assessment:

- Full Self-Assessment. This is common in the US. In full self-assessment, a taxpayer reports their income details on their tax return, calculates their tax per the tax return instructions, and files (and pays or expects a refund) by the due date of the filing period. If the tax return has errors or is changed during processing, the taxpayer receives an adjustment notice, but never receives an assessment notice.
- Partial Self-Assessment. This is common in taxes modeled after the British tax system. Like full self-assessment, the taxpayer reports their income tax details on a tax return. However, the tax return does not include instructions for calculating tax. When the tax return is processed, a Notice of Assessment is generated and sent to the taxpayer. The Notice of Assessment is a legal document informing the taxpayer of their tax obligation, and the due date for paying it.

The assessment established by the taxpayer's initial tax return is called the original assessment. If the taxpayer is later audited, and their income is increased (such that they now owe more tax), the additional tax established by the audit is part of an audit assessment. The incremental tax from the audit will be subject to different business rules, such as different rates and rules for calculating penalty and interest, and different collections notices or processes. It is also possible for a change to a tax return to result in a decrease in tax (and therefore a refund).

Bill Based Taxes

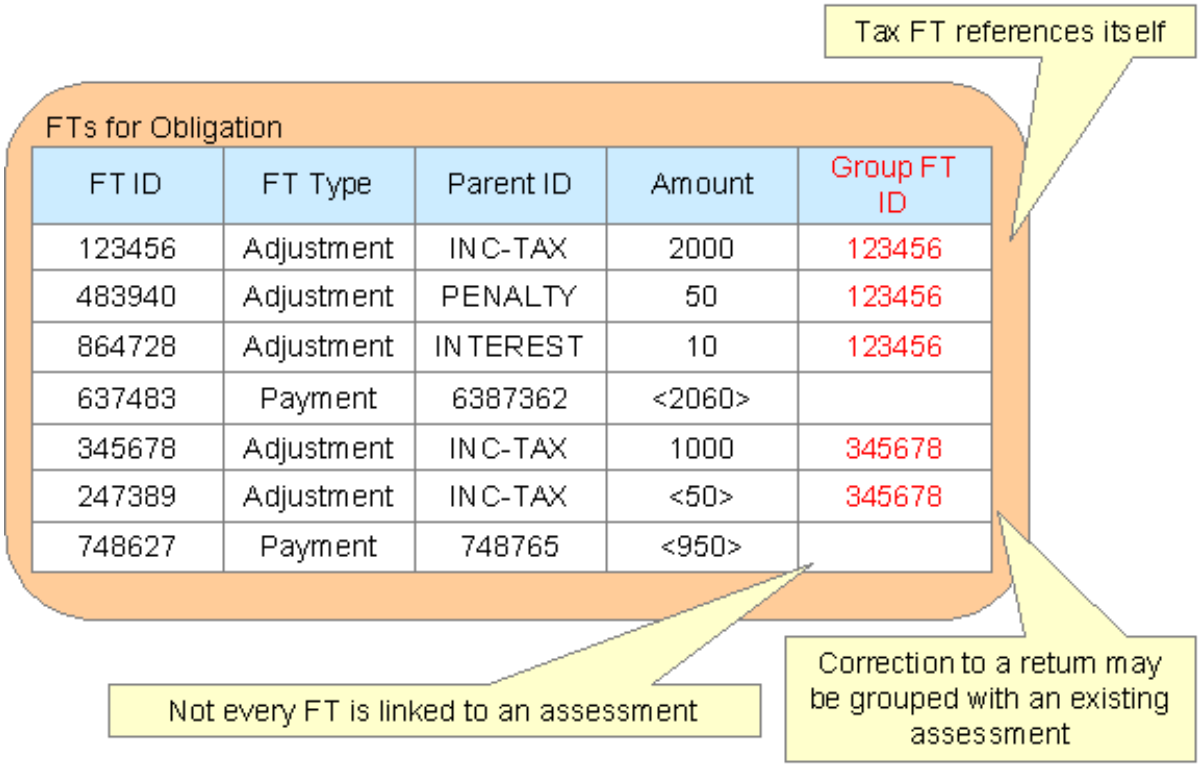
Bill based taxes are billed on a predefined schedule. Property tax, periodic licensing fees, and metered oil/resource taxes are examples of bill-based taxes.

For bill-based taxes, the system has all of the necessary information to initiate assessments. The system creates assessments on a schedule and a single assessment exists for each billing period. If a taxpayer disputes a bill, the bill is cancelled and rebilled without creating an incremental assessment.

Group Financial Transactions

As financial transactions are created for an obligation, often they are related to a specific liability such as an assessment, which is also a financial transaction (associated with an adjustment or a bill segment). For example, a payment may be made for a specific assessment. Penalties, interest and fees are often levied toward a specific assessment. Associating these subsequent financial transactions with the appropriate assessment FT ensures accurate penalty and interest calculations and accurate views of the balance of each assessment associated with an obligation.

The system provides a means for your implementation to group financial transactions (via the Group FT ID). When creating a financial transaction that should be associated with a given assessment, set the group FT id to the FT Id of the assessment. The following example shows the usage of the Group FT Id field.



The original assessment FT (sometimes referred to as the "header FT") references its own FT id in the Group FT field. This allows system functionality to easily identify the assessment financial transaction.

Effective Date

The effective date is the effective date of the payment, bill, or adjustment. For more information about effective dates and payment dates, see *Payment Date and Effective Date for Payment Events*. The effective date is used in calculating penalties and interest. For more information, refer to *The Big Picture of Penalty and Interest*.

Penalty and Interest

Calculating penalty and interest (P&I) for delinquent assessments is an important element of a tax authority's business.

Fastpath:

Refer to *The Big Picture of Penalty and Interest* for details about this functionality.

Bills, Payments & Adjustments

The following points are high level descriptions of the various financial transaction supported

- Adjustments are used for many different financial effects on a obligation, including tax assessments, penalties, interest, and fees. For more information about an adjustment, see *Adjustment Details*.
- Over time, many payments may be applied to an account's debt. For more information about a payment, see *Payment Details*.
- For bill based taxes, many bills are produced for an account over time. For more information about a bill, see *Bill Details*.

Note that the system maintains debt on each individual obligation for an account. An account's debt is the sum of its obligations' debt.

Bill Details

The following points highlight important concepts related to bills:

- A bill is produced for an account. Over time, many bills may be produced for an account.
- Bills contain bill segments. A bill segment is created to levy charges for a bill-based obligation. You may configure your system to generate a single bill for a single obligation. For example you may generate one bill for your property tax obligation's charges. You may also include bill segments for multiple obligations in a single bill. For example you may generate a bill that includes bill segments for all the personal property obligations.
- Bill segments contain calculation details. A bill segment contains information showing how the segment was calculated and how it should be printed on the taxpayer'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 obligation's current and payoff balances and on the general ledger.
- Canceling a bill cancels the financial transaction. 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 points highlight important concepts related to payments:

- A payment amount is ultimately directed towards an obligation via a payment segment.
- If an account has multiple obligations that are in debt, the various payment segments created for the obligations may be grouped into a single Payment for the account.
- A payment segment has a related financial transaction. A financial transaction contains the financial effects of the segment on the obligation's current and payoff balances and on the general ledger.
- Canceling a payment cancels the financial transaction. 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 points highlight important concepts related to adjustments

- Over time, an obligation may have many adjustments.
- An adjustment has a related financial transaction. The financial transaction contains the financial effects of the adjustment on the obligation's debt and on the general ledger.
- Canceling an adjustment cancels the financial transaction. 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 amount attributes: payoff amount and current amount. These attributes allow you to keep track separately of amounts related to how much the taxpayer owes.

- Current amount contains how much the taxpayer is asked to pay. In other words, this is the amount that affects the taxpayer's balance. It is what the taxpayer owes with respect to collections and is the amount that penalty and interest is calculated on.
- Payoff amount contains how much the taxpayer actually owes. This is the amount posted to the general ledger.

For most financial transactions, these values are the same. There are a small number of cases where this amount may be different. One example is a charitable contribution. If a taxpayer makes a charitable contribution when paying their tax liability, the payment or adjustment associated with the contribution credit should affect the payoff amount and the general ledger (because this amount is actually cash in hand). However, it should not affect the current amount because the contribution is not paying off any debt incurred and should not cause the taxpayer's balance to go into credit.

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 *Defining Bill Segment Types*.

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

Fastpath:

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

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

Fastpath:

For more information, refer to *Adjustments - Current Balance versus Payoff Balance* and *Setting Up Adjustment Types*.

GL Accounting Information

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

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.

Preventing Obligation 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 obligation's balance is affected. For example:

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

For payments, there is no issue. However, for bill based tax types, you may not want the taxpayer's balance to be impacted until the bill is completed. Consider the following:

- If a taxpayer has multiple obligations that should be presented on the same bill, it's possible for one of the obligations to have a bill segment that's in **error** and the other obligation's bill segment to be **frozen**.
- The **frozen** bill segment will impact the taxpayer's balance and the general ledger. 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 taxpayer's balance).

If this is unacceptable, you can setup the system to not allow certain types of FT's to be frozen until the bill is completed. This means that neither the taxpayer'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*.
- Determine if any of your *adjustment types* are ones that would be included on a bill-based tax bill. Select **Freeze At Bill Completion** for those that should not impact the taxpayer's balance or the general ledger until the next bill is completed. Select **Freeze At Will** for those that should impact the taxpayer's balance and the GL when they are frozen.

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.
- Any background process created for batch billing should not freeze bill segments until all segments on a bill are error free.
- Bill segments will exist in the **freezable** state until the bill is **completed**.
- If you turn on **Freeze At Bill Completion** for an adjustment type:
- Users will not be allowed to freeze adjustments of this type online.
- Background processes that create adjustments will not freeze this type of adjustment. Rather, the adjustments will be frozen when the next bill is completed.
- Adjustments of this type will therefore exist in the **freezable** state until the next bill is **completed**.

Note:

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

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

- If you turn on **Freeze At Will** on *Installation Options - Billing*:
- Users will be allowed to freeze bill segments online.
- Any background process created for batch billing should freeze bill segments when the individual segment is error-free.
- 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 also be set to **Freeze At Will** (otherwise they wouldn't get frozen).
- If you turn on **Freeze At Will** for an adjustment type:
- Users will be allowed to freeze adjustments of this type online.
- 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 an obligation
- Every obligation references an obligation type
- Every obligation 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).
- 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 obligations). 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.
- Any batch billing process should 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 CompletionBill 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 Obligation Balances And The GL From Being Impacted Until Completion* for more information.
- If you choose **Change if period is closed**:
- When a user freezes a bill segment online, they will only be prompted to supply an accounting date if the related accounting period is closed (because the accounting period closes after the bill segment is generated but before it's frozen). The current date will default, but the user can override this date.
- When a user freezes an adjustment online, they will only be prompted to supply an accounting date if the related accounting period is closed (because the accounting period closes after the adjustment is generated but before it's frozen). The current date will default, but the user can override this date.

Obligation 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 obligation level, not at the account level. This means that bills and payments are meaningless on their own. It's the obligations' bill segments, payment segments and adjustments that affect how much a taxpayer 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 taxpayer'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.

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

Note:

An obligation type controls numerous aspects of an obligation'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 obligation types. Refer to *Setting Up Obligation 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.

The Source Of GL Accounts On Financial Transactions

The following is a summary of the source of GL accounts on financial transactions:

- If a bill segment has a financial effect, the distribution code to debit comes from the distribution code on the obligation 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 obligation type.
- For adjustments that have a financial effect, refer to *Adjustment Type Defines the GL Account* for more information.

The following table lists some examples of 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 bill segment. *Bill Segment FT Algorithm is Payoff Amt = Bill Amt / Current Amt = Amt Due*	Debit: A/R	Obligation Type
	Credit: Revenue	Rate Component
Create a bill for company usage. *Bill Segment FT Algorithm is Payoff Amt = 0 / Current Amt = 0*	Debit: Company Usage Expense	Obligation Type
	Credit: Revenue	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 obligation	Debit: Cash	Bank Account on the Tender Source of the Tender Control for the Payment Segment's Tender.
	Credit: A/R	Obligation Type
Create a payment segment for a charitable contribution obligation	Debit: Cash	Bank Account on the Tender Source of the Tender Control for the Payment Segment's Tender.
	Credit: Charity Payable	Obligation Type
Create a payment segment for auto pay at bill completion time	Debit: Cash	Bank Account on the Tender Source on the Auto Pay Route Type of the Auto Pay Source.
	Credit: A/R	Obligation Type
Canceling a payment	Debit: A/R	Obligation 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	Obligation Type

Financial event	*GL Accounting*	*Source Of Distribution Code*
	Credit: Revenue	Adjustment Type
Create an adjustment with disbursement details	Debit: A/R	Obligation Type
	Credit: Revenue (multiple)	Adjustment calculation line details

General Financial Setup

The following sections describe maintenance transactions related to common financial objects.

Setting Up Divisions

There are two types of divisions referenced on an obligation type: a division and a GL division.

- General Ledger divisions typically comprise individual entities (e.g., companies) in your general ledger. You must set up a GL division for each such entity. The GL division's sole purpose in the system is to define the accounting period associated with financial transactions linked to obligations associated with the GL division (obligations are associated with GL divisions via their obligation type). The system cares about accounting periods in order to prevent a user from booking moneys to closed periods. It also uses accounting periods when it produces the flat file that contains the consolidated journal entry that is interfaced to your general ledger (refer to *The GL Interface* for more information).
- A division is used to separate business rules and can often be associated with a jurisdiction. The definition of a jurisdiction is a geographic-oriented entity with unique business rules. Another example of where you may use separate divisions is when your tax authority is responsible for other types of debt in the system, such as state university debts or child support debts. You must set up a division for each segmentation in your authority where business rules may differ.

Division is also referenced on obligation, location and account.

- The division on obligation is actually part of the obligation's obligation type. Because obligation type controls many business rules, all business rules that are on the obligation type can be thought of as being defined for a given jurisdiction and obligation type combination. Refer to *Defining Obligation Types* for more information.
- The division on location defines the jurisdiction in which the location is located. This jurisdiction controls the types of obligations that can be associated with the location.
- The division on account when combined with the account's account type defines the jurisdiction that governs financial business rules (e.g., the bill's due date). Refer to *Setting Up Account Types* for more information about these rules. The division on account can also play a part in the addressee of To Do entries associated with the account. To assign To Do entries to a role based on the division, simply link the To Do type to the division. Refer to *To Do Entries Reference A Role* for more information.

Note:

Both division and GL division are stored on the financial transactions associated with an obligation. However, only GL division plays a part in *The GL Interface*. Refer to *Setting Up GL Divisions* for information about GL Divisions.

The following topics describe the pages used to maintain a division.

Division - Main

To define a division, select choose **Admin Menu** > **Division**.

Description of Page

Enter an easily recognizable **Division** and **Description** for the division.

Enter the **Work Calendar** that defines the days on which this division operates. This calendar is used to ensure system-calculated dates (e.g., bill due date, credit and collection event dates, etc.) fall on a workday.

Use the **To Do Roles** scroll area if an account's division influences the role assigned to the To Do entries associated with the account. In the collection, define the **To Do Role** to be assigned to entries of a given **To Do Type** that are associated with accounts that reference the **Division**. Refer to *Assigning A To Do Role* for more information.

Note:

Only To Do entries that are account-oriented take advantage of the roles defined for a division.

Where Used

Follow this link to view the tables that reference *CI_CIS_DIVISION* in the data dictionary schema viewer.

Division - Characteristics

You can define characteristics for a division. You may need these for reporting purposes or in your algorithms. Refer to *Characteristic Types* for more information.

Open **Admin Menu** > **Division** and navigate to the **Characteristics** tab to maintain a division's characteristics.

Description of Page

Select a **Characteristic Type** and **Characteristic Value** to be associated with this division. Indicate the Effective Date of the characteristic type and value.

Note:

You can only choose characteristic types defined as permissible on a division record. Refer to *Setting Up Characteristic Types & Their Values* for more information.

Setting Up Revenue Classes

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

Debt Categories are used to categorize financial transactions that are debits. Debt categories may also be assigned to credit financial transactions if the credit is associated with a given type of debit. Refer to *Debt Categories and their Priorities* for more information.

To set up a debt category, open **Admin Menu** > **Debt Category**.

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

Debt Category List

The Debt Category *List zone* lists every debt category. The following functions are available:

- Click the *broadcast* icon to open other zones that contain more information about the adjacent debt category.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each debt category.

Click the **Add** link in the zone's title bar to add a new debt category.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Debt Category

The Debt Category zone contains display-only information about a debt category. This zone appears when a debt category has been broadcast from the Debt Category List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

Setting Up Debt Category Priorities

Debt category priorities allow you to define a priority order of allocating credit financial transactions to debit financial transactions during credit allocation. Refer to *Debt Categories and their Priorities* for more information.

To set up a debt category, open **Admin Menu** > **Debt Category Priority**.

The topics in this section describe the base-package zones that appear on the Debt Category Priority portal.

Debt Category Priority List

The Debt Category Priority *List zone* lists every debt category priority. The following functions are available:

- Click the *broadcast* icon to open other zones that contain more information about the adjacent debt category priority.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each debt category priority.

Click the **Add** link in the zone's title bar to add a new debt category priority.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Debt Category Priority

The Debt Category Priority zone contains display-only information about a debt category priority. This zone appears when a debt category priority has been broadcast from the Debt Category Priority List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

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* 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** provides the ability to define configuration to write off debt associated with the distribution code by transferring the written off debt to a separate obligation. You would only do this if you wanted to the written off debt to remain visible in the account's balance and you don't want it to remain in the original obligation's balance.

- Define the **Division** and **Obligation Type** of the obligation to which bad debt associated with this distribution code should be transferred at write-off time. Note: only obligation types with a special role of **Write Off** may be selected.
- When the system transfers debt to the write-off obligation defined above, the distribution code defined on this **Division / Obligation 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.

Note:

The write off algorithms provided in the base product (as part of overdue processing) do not transfer the debt to a separate **write off** obligation.

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.
- By default, the installation is configured to practice *fund accounting*. With this option activated, you can define the **Fund** associated with this distribution code. If your installation options indicate that fund accounting is **not practiced**, the field is not visible.
- Use the grid to define characteristic values for the **Distribution Code**. To modify a characteristic, simply move to a field and change its value. The following fields display:

- **Characteristic Type**. Indicate the type of characteristic.
- **Characteristic Value**. Indicate the value of the characteristic.

Note:

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

Where Used

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

Managing Adjustment Setup

An obligation's debt may be changed with an adjustment. Every adjustment must reference an adjustment type. The adjustment type contains a great deal of information that is defaulted onto the adjustment, including whether the adjustment amount is calculated. It also controls many business processes associated with the adjustment. The topics in this section describe how to design and set up adjustment types.

Adjustment Types Define Business Rules

An adjustment type contains the business rules that govern how its adjustments are managed by the system. The topics in this section describe how adjustment type controls the behavior of an adjustment.

Defines the Business Object

The adjustment user interface relies on a business object to define appropriate UI maps for display and maintenance. In addition, the business object may be used to define additional business rules. The adjustment's business object is defined on the adjustment type.

Note:

The adjustment maintenance object does not have a flexible lifecycle so the adjustment's business object may not be used to define a lifecycle or lifecycle rules.

The base product supplies several business objects based on broad categories of adjustments. To view the details of the business objects provided by the base product, navigate using **Admin Menu** > **Business Object** and select the business objects for the **ADJUSTMENT** maintenance object.

If the base business objects do not provide all the desired functionality for capturing and displaying your adjustment information, you may extend the base business objects as desired or create new ones that are appropriate for your business.

Defaults the Adjustment Amount

The adjustment type may default the adjustment amount in one of the following ways:

- The adjustment type may specify a default amount. This would be used for those adjustment types that have a standard charge for all taxpayers that receive this adjustment.
- The adjustment type may specify a default adjustment amount algorithm. This would be used for those adjustment types that have a charge that varies based on other factors. For example, a non-sufficient funds charge may be based on a taxpayer's credit rating.

When an amount is defaulted onto a new adjustment it may be overridden by a user.

Generated Adjustments

You can use an algorithm to calculate an adjustment amount or to build details related to the adjustment amount. The following are some examples of where this may be used:

- Taking a base adjustment amount and applying a rate to add additional charges.
- Taking the adjustment amount and capturing or producing additional supporting details in the calculation details collection. For example, if a P&I adjustment should be "disbursed" into detailed buckets as per the disbursement of the related assessment adjustment, a generate algorithm may be used to produce the appropriate P&I calculation details. Refer to the base product algorithm type **C1-PIDISTRII** for an example.
- Receiving calculation details from a calling program and storing them with the adjustment.
- Receiving calculation details from a calling program and using them to generate general ledger details for the adjustment's financial transaction.

All generated adjustment types must be set up as follows:

- Set the adjustment type's Adjustment Amount Type to **Calculated Amount**
- Plug in an appropriate Generate algorithm on the adjustment type as per the business rules
- Configure an appropriate FT creation algorithm on the adjustment type. A typical reason for using generated adjustments is that additional details are required for the general ledger. The base product FT creation algorithms all include an option to use the calculation details as a source for the GL.

Adjustment Calculation Details

Generated adjustments are used to produce details related to the adjustment amount. These details may be provided by a calling program or may be determined using the appropriate business logic.

In general, the adjustment calculation lines are used to capture the details. But there are unusual points related to this logic:

- Algorithms that populate the calculation line details must also populate calculation "header" details, including the number of lines and the total amount. However, calculation "header" details are never instantiated in the database. The information populated in the calculation header is used by the adjustment logic to process the calculation lines. Refer to any base product Generate algorithm for an example of populating the calculation "header".
- Calculation lines include a switch called "create bill line." (Adjustment calculation lines are reusing bill calculation line functionality). In the adjustment logic, adjustment calculation lines are only instantiated if this switch is set to true. The reason that an algorithm may produce a calculation line that should not be stored is that this information is available in memory for subsequent processes, namely the FT creation algorithm, which produces the GL details.

There are times when the calling program has the detailed information to be stored in the adjustment calculation lines and in the FT GL details. The services provided in the product to add and freeze an adjustment do not include the full

adjustment calculation details collection or the FT GL details collection as input. Rather, there is a special field called "custom common area" which may be used to pass information in XML format into the adjustment routines. The base product provides a Generate algorithm that accepts calculation details passed in the custom common area and returns calculation lines. Refer to the base product algorithm type **C1-ADJ-GN-CL** for more information.

Applying a Rate

One use of the adjustment generation plug-in is to call rate application. The base product supplies an algorithm type **ADJG-RT** that enables you to call a rate application, passing in the base adjustment amount and receiving calculated details from the rate.

The adjustment type's generate adjustment algorithm controls which rate is applied to the base amount. A user supplied calculation date controls which version of the rate is used. The user may supply the base amount or it may be defaulted from the adjustment type and possibly overridden by the user prior to calculating the adjustment amount.

Controls Which Balance(s) Are Affected

The adjustment type's financial transaction (FT) algorithm controls how payoff balance and current balance are affected by the adjustment amount.

Defines the GL Account Affected by the Adjustment

Most adjustments affect the general ledger (GL) in some way. The following points describe the source of these GL accounts.

- For many adjustments there is a single accounting entry generated:
- One side of the accounting entry is taken from the distribution code on the obligation type of the obligation affected by the adjustment. For example, if you are adjusting the payoff balance on a normal obligation, the A/R account is constructed from the distribution code on the obligation's obligation type.
- The other side of the accounting entry is taken from the distribution code on the adjustment's adjustment type.
- For transfer adjustments (i.e., adjustments used to transfer moneys between two obligations), there are two accounting entries generated - one for the "from" side and one for the "to" side. Each adjustment carries its own set of balanced GL accounting details.
- For each adjustment, one side of the entry is taken from the distribution code on the obligation type of the obligation affected by the adjustment
- The other sides of both accounting entries have the same GL account. This account should be the intermediate clearing GL account that is to be used for the transfer. The source of this clearing GL account is the distribution code on the adjustment type used to transfer the funds.
- For generated adjustments, the accounting entry may include several GL details:
- One side of the entry is taken from the distribution code on the obligation type of the obligation affected by the adjustment.
- The other side of the entry depends on configuration on the adjustment financial transaction algorithm. If it is configured to use the Calculation Lines as the source, the distribution codes are taken from the calculation lines. If it is configured to use the adjustment type as the distribution code source, the other side of the accounting entry is taken from the distribution code on the adjustment type.

Note:

Not all adjustments affect the GL. As a general rule of thumb, only those adjustments that affect an obligation's payoff balance affect the GL.

Controls the Interface to A/P and Income Statement Reporting

If the adjustment type is associated with a payment of money to a taxpayer (e.g. a refund), the adjustment type indicates such with a reference to an A/P request type.

When an adjustment that references an A/P request type is **frozen**, an A/P download request record is created. This record is the interface request to ask your A/P system to cut a check. This interface record is marked with a batch process ID and run number.

- The batch process ID is the process responsible for creating the flat file that contains check request that is interfaced to your account's payable system. The batch process ID is defined on *Installation Options - Financial*. The base package is supplied with a skeletal background process (referred to by the process ID of **APDL**) that must be populated with logic to format the records in the format compatible with your accounts payable system.
- The run number is the batch process ID's current run number.

Fastpath:

Refer to *Accounts Payable Check Request* for more information.

If the resultant check needs to be reported for income tax purposes under a specific income statement category, the category is also specified on the adjustment type. The income statement category is in turn interfaced to the A/P system (the system does NOT manage income statement reporting).

Controls Information Printed On the Bill

If the adjustment is one that appears on a taxpayer's bill, the verbiage is specified on the adjustment type (and may NOT be overridden on the adjustment).

Controls if the Adjustment Requires Approval

Fastpath:

Refer to *The Big Picture of Adjustment Approvals* for more information.

May Control the Adjustment Information

The adjustment information displayed throughout the system is controlled by a plug-in.

The system first looks to see if the adjustment type references an adjustment business object and if the business object defines an information plug-in. If a BO is not provided or if that BO does not define an information algorithm, the system looks for an information algorithm plugged into the adjustment maintenance object.

If no plug-ins are found on the BO or the MO, the system looks for a plug-in algorithm on the *Adjustment Type*.

If such an algorithm is not plugged-in on the Adjustment Type, the system looks for a corresponding algorithm on the *installation record*.

If you prefer different formatting logic, your implementation should provide a plug-in at one of the above plug-in spots, as appropriate for your business rules.

Setting Up Adjustment Types

The topics in this section describe how to set up adjustment types.

Note:

When a new adjustment type is added. When you introduce a new adjustment type, you must update one or more adjustment profiles with the new adjustment type. This is because adjustment profiles define the adjustment types that may be levied on obligations (adjustment profiles are defined on obligation 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.

If an *adjustment type extension* exists for the adjustment type, a link to the extension record is displayed next to the Adjustment Type ID.

The **Adjustment Amount Type** indicates whether or not the adjustment amount or details to support the adjustment amount is generated or not. Select **Calculated Amount** when you want to use a generate algorithm to generate the adjustment amount or generate calculation line details to support the adjustment amount, otherwise select **Non-Calculated Amount**. Refer to *Generated Adjustments* for more information about generated 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 obligation type.

Note:

Distribution Code for Generated Adjustments. Depending on the algorithm used for the *generated adjustment*, the distribution code may come from the adjustment type or the calculation lines of the algorithm. If the adjustment's FT creation 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*.

Use **Adjustment Type Category** to assign a broad category to the adjustment type. Valid values are **Assessment**, **Credit**, **Manual Adjustment**, **Manual P&I**, **Penalty & Interest**, **Waiver**, **Write Off**.

Fastpath:

Used by P&I. The values of **Assessment**, **Penalty & Interest** and **Waiver** play an important role in the base product P&I Calculation algorithm. Refer to *Apply P&I Rules for Each Time Period*for more details.

Note:

Multi-Adjustment Transactions The values of **Manual Adjustment** and **Manual P&I** control whether the adjustments of this type may be created on the Manual Adjustment and Manual P&I transactions respectively.

Note:

The values for this field are customizable using the Lookup table. This field name is ADJ_TYPE_CAT_FLG.

Define the **Debt Category** to assign to the financial transaction for debit adjustments (adjustments with amounts greater than or equal to 0) created for this adjustment type.

Note:

Required for base algorithms. The base P&I calculation algorithm and the base *Determine Detailed Balance* algorithm require that every debit adjustment reference a debt category.

If this adjustment type is for a certain type of credit adjustment and it has special rules for how credits are applied in the base Determine Detailed Balance algorithm, define the appropriate **Debt Category Priority**. Refer to *Debt Categories and their Priorities* for more information.

Enter the **Currency Code** for adjustments of this type.

Turn on **Sync. Current Amount** if adjustments of this type exist to force an obligation's current balance to equal its payoff balance. These types of adjustments are issued before an obligation's funds are transferred to a write-off obligation. If this switch is on, choose an **Adjustment Fin Algorithm** that does not impact payoff balance or the GL, but does affect the obligation's current balance (refer to *ADJT-CA* for an example of such an algorithm).

Enter a **Default Amount** if an amount should be *Default the Adjustment Amount* onto adjustments of this type.

Fastpath:

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

If the A/P Adjustment should be recorded in respect of the taxpayer's income statement amounts, indicate the **Income Statement**. The values of this field are **Interest** and **Miscellaneous**. This type of adjustment would also have an **A/P Request Type Code** selected, as income statement reporting is handled in A/P.

Turn on **Print By Default** if information about adjustments of this type should print on the account's next bill.

Choose an **A/P Request Type** if this adjustment is interfaced to accounts payable (i.e., it's used to send a refund check to a taxpayer). Refer to *A/P Check Request* for more information.

The **Adjustment Freeze Option** defines when adjustments can be frozen and therefore when an obligation's balance and the general ledger are affected by an adjustment. Refer to *Preventing Obligation 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.

Set **Allow Manual Creation**to **Allowed** if you want to allow users to manually create adjustments of this type from the adjustment page, otherwise set it to **Not Allowed**.

Set **Allow Manual Cancellation** to **Allowed** if you want to allow users to manually cancel adjustments of this type from the adjustment page, otherwise set it to **Not Allowed**.

If adjustments of this type require approval, define an **Approval Profile**. For more information, refer to *The Big Picture of Adjustment Approvals*.

Enter the verbiage to appear on the printed bill in **Description on Bill**.

Use an **Adjustment Business Object** to define a *BO* that may govern the display and maintenance UI maps for adjustments of this type as well as additional rules related to adjustments of this type.

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

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.
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" may be constructed using an algorithm plugged in here. Refer to *Adjustment Information* for more information on when this algorithm is used.

		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 obligation's payoff and current balances. It also constructs the information that is eventually interfaced to your general ledger. Click *here* to see the algorithm types available for this system event.
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 Obligation	Optional	Algorithms of this type are used to find an obligation for which the adjustment can be posted. This plug-in is used particularly during adjustment upload when a staging record does not identify the obligation 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 generate the adjustment amount or generate details to support the adjustment amount if an adjustment type indicates that the adjustment amount is calculated. Refer to *Generated Adjustment* 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.

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 an obligation type, you limit the type of adjustments to be linked to the obligation type's obligations. The creation of adjustment profiles and their linkage to obligation types prevents inappropriate adjustments from being linked to your obligations. More than one adjustment type profile may be linked to an obligation 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 obligation types that are allowed to levy such fees.

Note:

Bottom line. An adjustment can only be linked to an obligation if its adjustment type is part of an adjustment type profile that is valid for the obligation's obligation 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.

Where Used

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

Setting Up Adjustment Type Extensions

Adjustment Type Extensions let you maintain additional information related to an adjustment type. The information you maintain is controlled by the business objects that your implementation defines based on your business needs.

The base product supplies an adjustment type extension business object that supports mapping between assessment distribution codes and P&I distribution codes. Refer to the business object **C1-PIDisbursement** for more information.

To set up an Adjustment Type Extension, open **Admin Menu** > **Adjustment Type Extension** .

The topics in this section describe the base-package zones that appear on the Adjustment Type Extension portal.

Adjustment Type Extension List

The Adjustment Type Extension *List zone* lists every Adjustment Type Extension. The following functions are available:

Click the *broadcast* icon to open other zones that contain more information about the adjacent Adjustment Type Extension.

The standard actions of **Edit, Delete** and **Duplicate** are available for each Adjustment Type Extension.

Click the **Add** link in the zone's title bar to add a new Adjustment Type Extension.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Adjustment Type Extension

The Adjustment Type Extension zone contains display-only information about an Adjustment Type Extension. This zone appears when an Adjustment Type Extension has been broadcast from the Adjustment Type Extension List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

Setting Up Adjustment Creation Reasons

Adjustment creation reasons are defined using an extendable lookup. To view or create adjustment creation reasons:

- Open **Admin Menu** > **Extended Lookup**.
- Search for and select the **Adjustment Creation Reason** extended lookup business object.
- The list of existing adjustment creation reasons are displayed in a standard *List zone*.
- Choose an existing adjustment creation reason to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new adjustment creation reason.

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.

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 and Income Statement Reporting* 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 **Payment Method**. Choose from these options:**System Check**

Where Used

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

The Big Picture Of Adjustment Approval

Some implementations require adjustments to be approved by one or more managers before they impact an obligation's balance and the general ledger. For example, an adjustment used to refund a credit balance may require managerial approval before the refund is sent to the taxpayer. 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 an obligation's debt. If an adjustment type references an approval profile, the approval profile's approval hierarchy defines if the adjustment requires approval and who the authorized approvers are. For example, an approval profile can be configured with the following approval hierarchy:

- Adjustments < $0 require approval by the "credit approvers role"
- Adjustments >= $0 and <= $10 do not require approval
- Adjustments > $10 and <= $100 require the approval of a user that belongs to the "level 1 approvers role"
- Adjustments > $100 require two levels of approval: first a user that belongs to the "level 1 approvers role" must approve the adjustment; afterwards, the adjustment must be approved by a user that belongs to the "level 2 approvers role"

Note:

Transfer adjustments. The term "transfer adjustment" refers to two adjustments that are used to transfer moneys between two obligations. 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 obligation's balance. If an adjustment's adjustment type references an approval profile, the user cannot freeze the adjustment directly. Rather, the user must submit the adjustment for approval when it's ready and only when the last applicable approver approves the adjustment will it become **Frozen**.

Note:

Freeze during bill completion. You can configure the system to only freeze certain types of adjustments when the next bill is completed for the adjustment's account. When the last approver approves such adjustments, they remain in the **Freezable**. When the next bill is completed for the account, these adjustment become **Frozen**. Such adjustments that have not been approved at the time of bill completion will remain in the **Freezable** state.

Refer to *Preventing Obligation Balances And The GL From Being Impacted Until Bill Completion* for more information.

Approval Requests Manage And Audit The Approval Process

Users submit an adjustment for approval using a dedicated button on the *Adjustment* page. When an adjustment is submitted for approval, the system creates an "approval request". The approval request determines if the adjustment requires approval and, if so, the list of approvers. If the adjustment does not require approval, the approval request is updated to indicate such and the adjustment is **Frozen** immediately (if freezing is allowed prior to bill completion). If the adjustment requires approval, the approval request's state becomes **Approval In Progress** and the approver(s) are notified.

Note:

Approval submission logic is customizable. The previous paragraph describes how the base-package works when an adjustment is submitted for approval. This logic resides in an algorithm that's plugged in on the **C1-AdjustmentApprovalProfile** business object in the **Determine Approval Requirements** system event. Your implementation can change this logic by developing a new algorithm and plugging it into this business object. If your logic is meant to supersede the base-package algorithm, remember to inactivate the base-package algorithm by adding an appropriate inactivation option to this business object.

To Do Entries Are Created To Notify Approvers

When an approval request detects an adjustment requires approval, it notifies the first approver by creating a To Do entry. The To Do entry is created using the To Do type and To Do role defined on the approval profile. All users who belong to the approving To Do role can see the entry. When a user drills down on an adjustment approval To Do entry, the *Adjustments - Approval* portal is opened. This portal contains summary information about the adjustment and the approval history of the adjustment. This portal is also where the user approves or rejects the adjustment.

When the first user in the To Do role approves an adjustment, the To Do entry is **Completed** and the approval request's audit log is updated. If there are no higher levels of approval required, the adjustment is **Frozen** (if freezing is allowed prior to bill completion) and the approval request is moved to the **Approved** state. If there are higher levels of approval required, a new To Do entry is created to the next To Do role in the approval hierarchy.

Note:

To Do entries can create email. A To Do entry can be configured to create an email message for every user in the To Do role to inform the user(s) of new adjustments requiring their attention. Refer to *To Do Entries May Be Routed Out Of The System* for the details.

Monitoring and Escalating Approval Requests

The base-package is supplied with an algorithm that your implementation can use to monitor approval requests that have been waiting too long for approval. This algorithm can complete the current To Do entry and create a new one for a different role when the time out threshold defined on the algorithm's parameters is exceeded. If you've configured the system to send email for approval, this algorithm can also send x reminder emails (where x is defined on the algorithm's parameters) before the approval request is escalated to the new To Do role. Refer to *C1-APR-TMOUT* for more information about this algorithm. If you plan to enable this functionality, plug-in your configured algorithm on the **Approval In Progress** state on the **C1-AdjustmentApprovalRequest** business object.

Rejecting Deletes The Adjustment

When an adjustment is being approved, anyone with access to the adjustment can reject it by using the *Adjustments - Approval* portal. Users other than the current approver are allowed to reject an adjustment to allow an "in process" an adjustment to be withdrawn.

When an adjustment is rejected, the following takes place:

- The user is prompted for a reject reason.
- The approval request's audit log is updated with the reject reason and the approval request is moved to the **Rejected** state.
- The adjustment is deleted.

Designing Your Approval Profiles

The following points describe a recommended design process:

- Create logical groups of adjustment types where each group has the same monetary hierarchy and approvers. An approval profile will be required for each of these groups.
- The number of To Do types (if any) that need to be created is dependent on how the adjustment approval To Do entries should be organized on To Do lists. For example, if all approval request To Do entries can appear in the same To Do list, you can use the base-package adjustment approval To Do type. However, if your organization prefers each approval profile's To Do entries to appear in a distinct To Do list, a separate To Do type will be needed for each list. Note that the base-package is supplied with a To Do type called *C1-ADAPP* that should be used as the basis for any new approval request To Do type.
- The number of To Do roles is dependent on who approves your adjustments. At a minimum, you will require a separate To Do role for each level in your approval profiles. Remember that every user in a To Do role will see its entries (and receive email if you've configured the system to do such).
- Refer to *Monitoring and Escalating Approval Requests* for how to configure the system to escalate approval requests that have been waiting too long.
- If your implementation requires email notification when an adjustment requires approval, the following setup is required:
  - Set up an outbound message type, external system, and 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 you can explore the physical tables and data relationships behind the approval functionality:

- 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 different business objects with their own business rules.

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** .

Fastpath:

Refer to *The Big Picture Of Adjustment Approval* for a detailed description of how approval profiles govern the adjustment approval process.

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

Approval Profile List

The Approval Profile *List zone* lists every approval profile. The following functions are available:

- Click the *broadcast* icon to open other zones that contain more information about the adjacent approval profile.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each approval profile.

Click the **Add** link in the zone's title bar to add a new approval profile.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Approval Profile

The Approval Profile zone contains display-only information about an approval profile. This zone appears when an approval profile has been broadcast from the Approval Profile List zone or if this portal is opened via a drill down from another page.

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

Approval Profile's Adjustment Types

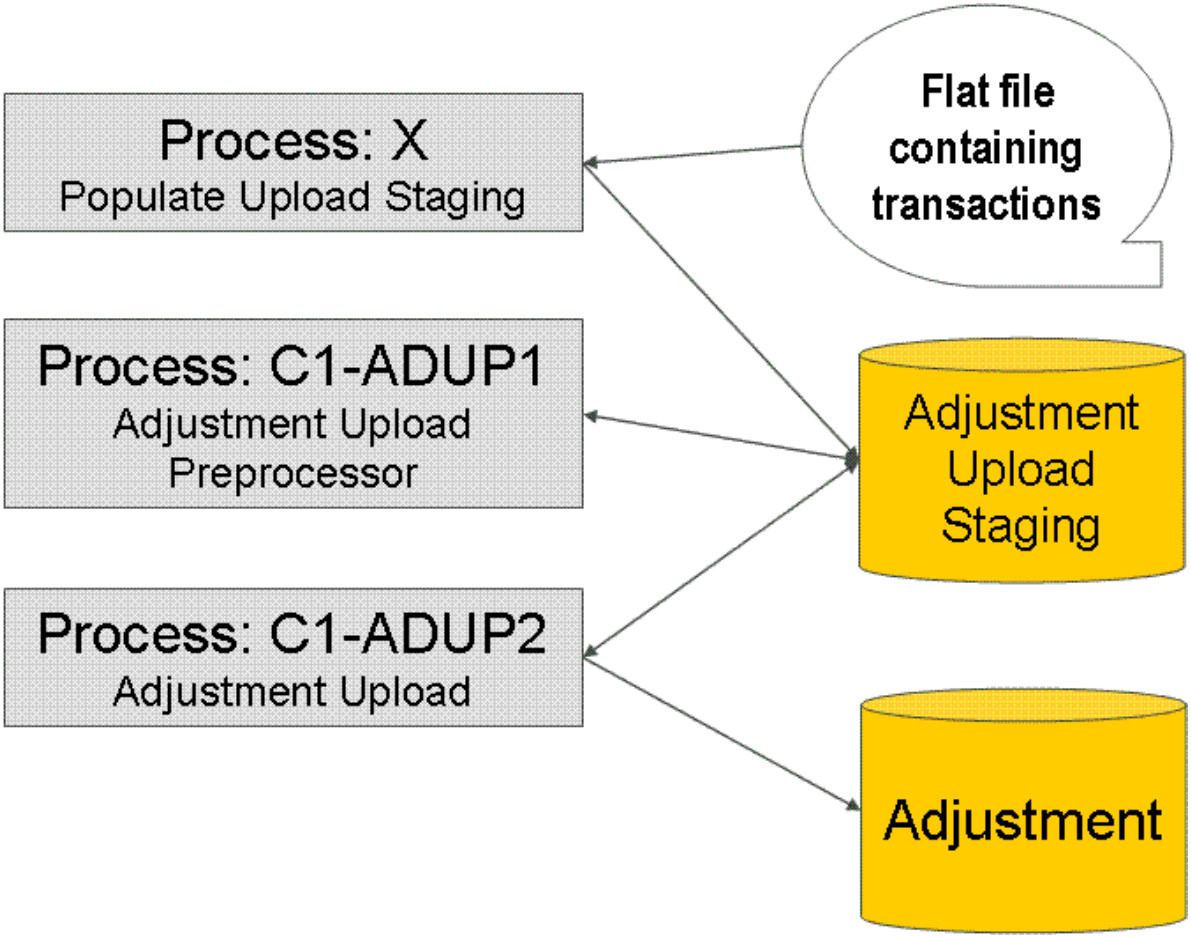
The Approval Profile's Adjustment Types zone lists every *adjustment type* that is governed by this approval profile. This zone appears when there is at least one adjustment type governed by the approval profile displayed in the Approval Profile zone.

Interfacing Adjustments From External Sources

The topics in this section describe how adjustments are uploaded from an external source.

Interfacing Adjustments

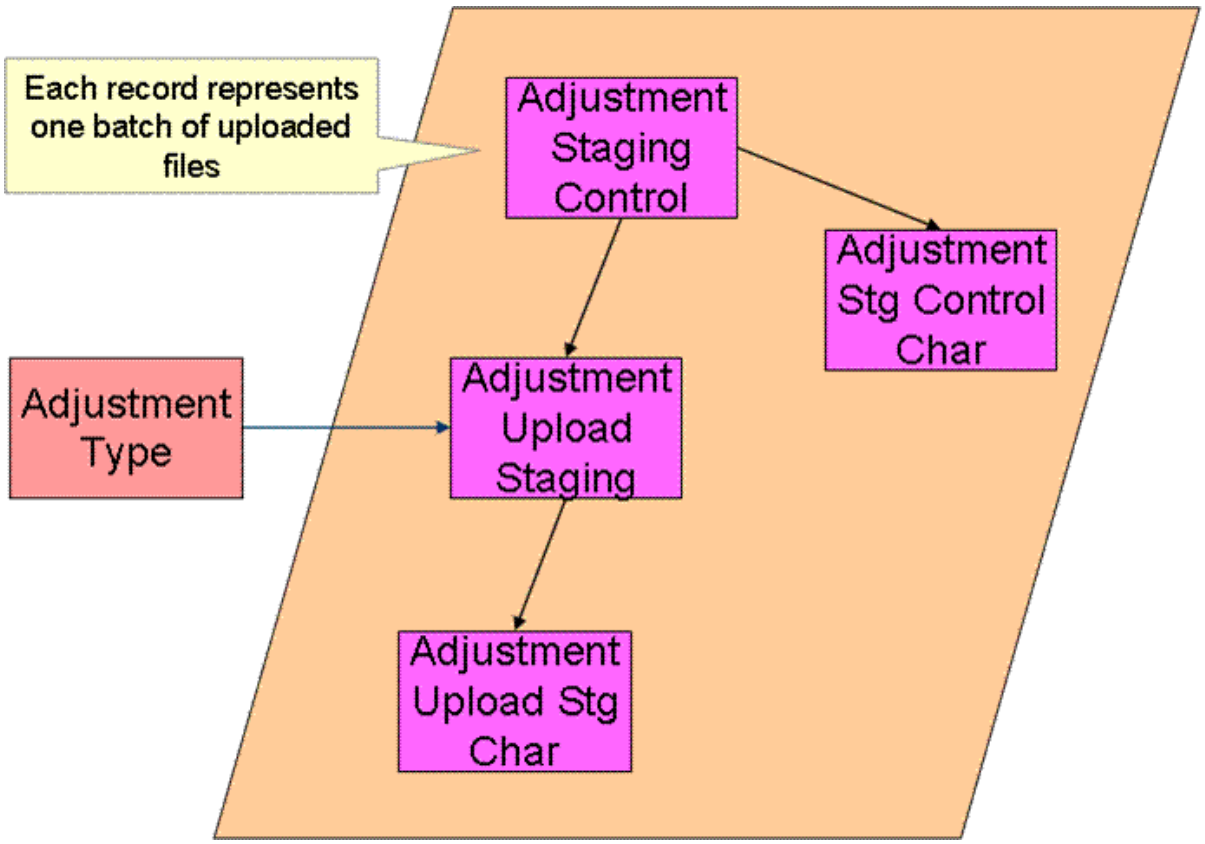
The following diagram illustrates the processes involved in the uploading of adjustments into the system.



The topics in this section describe how these processes work.

Process X - Populate Adjustment Upload Records

Process X refers to the mechanism used by your organization to populate the various staging tables (shown in the shaded section of the following ERD).



The topics in this section describe each of these tables.

Adjustment Staging Control

You must create an adjustment staging control record for each batch of adjustments to be uploaded into the system. The name of this table is CI_ADJ_STG_CTL. The following describes each column on this table.

Column Name	*Length*	*Req'd*	*Data Type*	*Comments*
ADJ_STG_CTL_ID	12	Y	N	This is the unique identifier of the adjustment staging control record. This key is a sequential number so you can use a database function to assign the value when populating the table.
CRE_DTTM	15	Y	DateTime	This is the date / time on which the adjustment staging control record was created. This must be populated with the current date / time.
ADJ_STG_CTL_STATUS_FLG	4	Y	A/N	This must be set to **P** for **Pending**.
ADJ_STG_UP_REC_CNT	10	Y	N	This is the total number of adjustment upload staging records that are

				linked to this adjustment staging control.
TOT_ADJ_AMT	13.2	Y	N	This column must equal the sum of adjustment amounts on the adjustment upload staging records that are linked to this adjustment staging control.
CURRENCY_CD	3	Y	A/N	This must be a valid currency code in the system. Refer to *Defining Currency Codes* for more information.
MESSAGE_CATEGORY	5	N	N	Leave this blank. The adjustment upload preprocessor populates this when an error occurs during upload.
MESSAGE_NBR	5	N	N	Leave this blank. The adjustment upload preprocessor populates this when an error occurs during upload.

Adjustment Staging Control Characteristic

You must create an adjustment staging control characteristic record for each characteristic that you would like to link to the adjustment staging control. The name of this table is CI_ADJ_STG_CTL_CHAR. The following describes each column on this table.

Column Name	***Length***	***Req'd***	***Data Type***	***Comments***
ADJ_STG_CTL_ID	12	Y	N	This must correspond with the prime key of the related CI_ADJ_STG_CTL record.
CHAR_TYPE_CD	8	Y	A/N	This must correspond with a characteristic type that is defined as valid for **adjustment staging control**. Refer to *Setting Up Characteristic Types & Their Values* for more information.
SEQ_NUM	3	Y	N	This should be set to **10** unless you have multiple values for a given adjustment staging control and characteristic type.
CHAR_VAL	16	N	A/N	Populate this field if your *characteristic type* is **predefined**.

ADHOC_CHAR_VAL	254	N	A/N	Populate this field if your *characteristic type* is **ad-hoc** or **file location**.
CHAR_VAL_FK1 - CHAR_VAL_FK5	50 each	N	A/N	Populate these fields if your *characteristic type* is **foreign key reference**. Up to five columns of 50 bytes each are provided to accommodate compound keys.

Adjustment Upload Staging

You must create an adjustment upload staging record for each adjustment you want to upload. The name of this table is CI_ADJ_STG_UP. The following describes each column on this table.

Column Name	***Length***	***Req'd***	***Data Type***	***Comments***
ADJ_STG_UP_ID	12	Y	N	This is the unique identifier of the adjustment upload staging record. This key is a sequential number so you can use a database function to assign the value when populating the table.
ADJ_STG_CTL_ID	12	Y	N	The ID of the adjustment staging control that is linked to this adjustment upload staging record.
ADJ_TYPE_CD	8	Y	A/N	This must correspond to the prime key of one of your adjustment types. Refer to *Setting up Adjustment Types* for more information.
ADJ_STG_UP_STATUS_FLG	1	Y	A/N	This must be set to **P** for **Pending**.
CREATE_DT	10	Y	Date	The date when the adjustment occurred.
ADJ_AMT	13.2	Y	N	The amount of the adjustment.
ADJ_SUSPENSE_FLG	4	Y	N	This must be set to **NSUS** for **Not In Suspense**.
SA_ID	10	N	A/N	This must correspond to a valid obligation in the system. If you leave this blank and opt to let the system find the obligation during adjustment

				upload preprocessing, a **Determine Obligation** algorithm must be plugged in on the associated adjustment type. This plug-in is meant to derive the obligation ID based on supplied miscellaneous information - e.g. information supplied via adjustment characteristic upload staging.
ADJ_ID	10	N	A/N	Leave this blank. The adjustment upload process populates this.
SUSPENSE_ADJ_ID	10	N	A/N	Leave this blank. The adjustment upload process populates this.
MESSAGE_CATEGORY	5	N	N	Leave this blank. The adjustment upload background processes populate this when an error occurs during upload.
MESSAGE_NBR	5	N	N	Leave this blank. The adjustment upload background processes populate this when an error occurs during upload.

Adjustment Characteristic Upload Staging

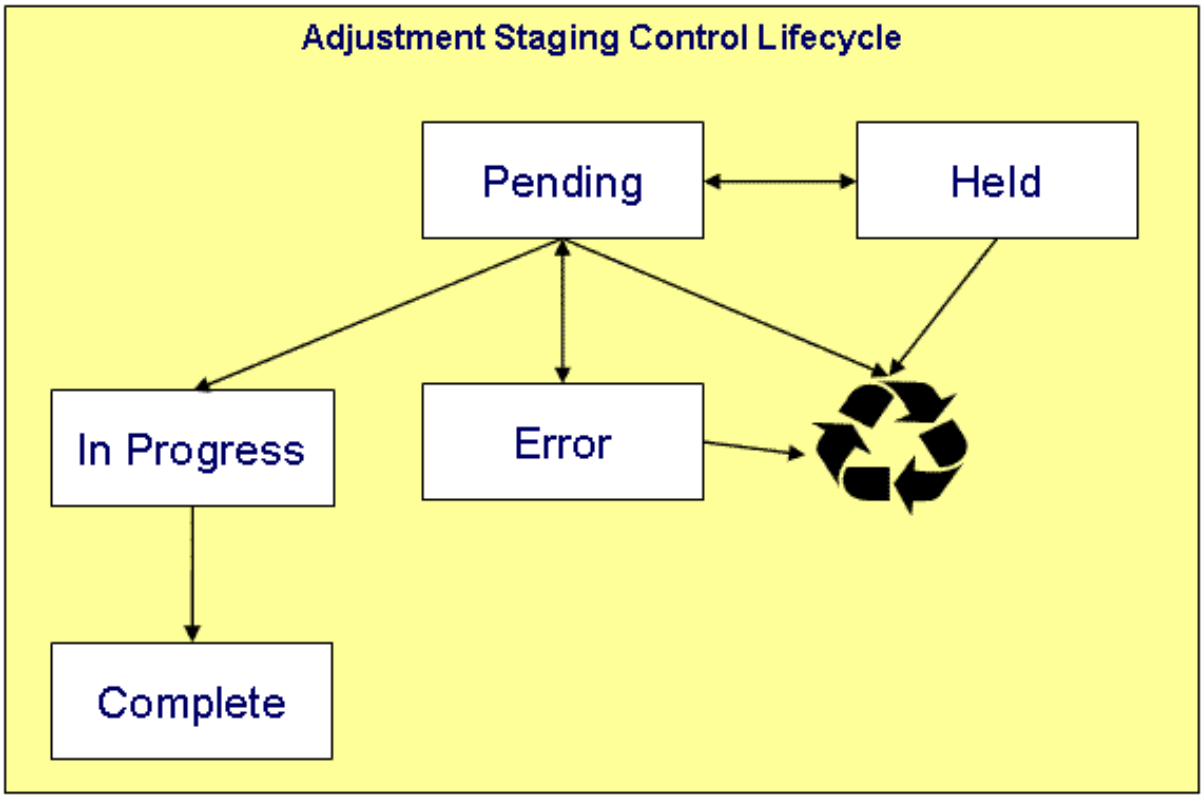
You must create an adjustment characteristic upload staging record for each characteristic that you would like to link to the adjustment upload staging. The name of this table is CI_ADJ_STG_UP_CHAR. The following describes each column on this table.

Column Name	***Length***	***Req'd***	***Data Type***	***Comments***
ADJ_STG_UP_ID	12	Y	N	This must correspond with the prime key of the related CI_ADJ_STG_UP record.
CHAR_TYPE_CD	8	Y	A/N	This must correspond with a characteristic type that is defined as valid for **adjustment** Refer to *Setting Up Characteristic Types & Their Values* for more information.
SEQ_NUM	3	Y	N	This should be set to **10** unless you have multiple values for a given adjustment

				upload staging and characteristic type.
CHAR_VAL	16	N	A/N	Populate this field if your *characteristic type* is **predefined**.
ADHOC_CHAR_VAL	254	N	A/N	Populate this field if your *characteristic type* is **ad-hoc** or **file location**.
CHAR_VAL_FK1 - CHAR_VAL_FK5	50 each	N	A/N	Populate these fields if your *characteristic type* is **foreign key reference**. Up to five columns of 50 bytes each are provided to accommodate compound keys.

The Lifecycle of an Adjustment Staging Control Record

The following diagram shows the possible lifecycle of an adjustment staging control record.

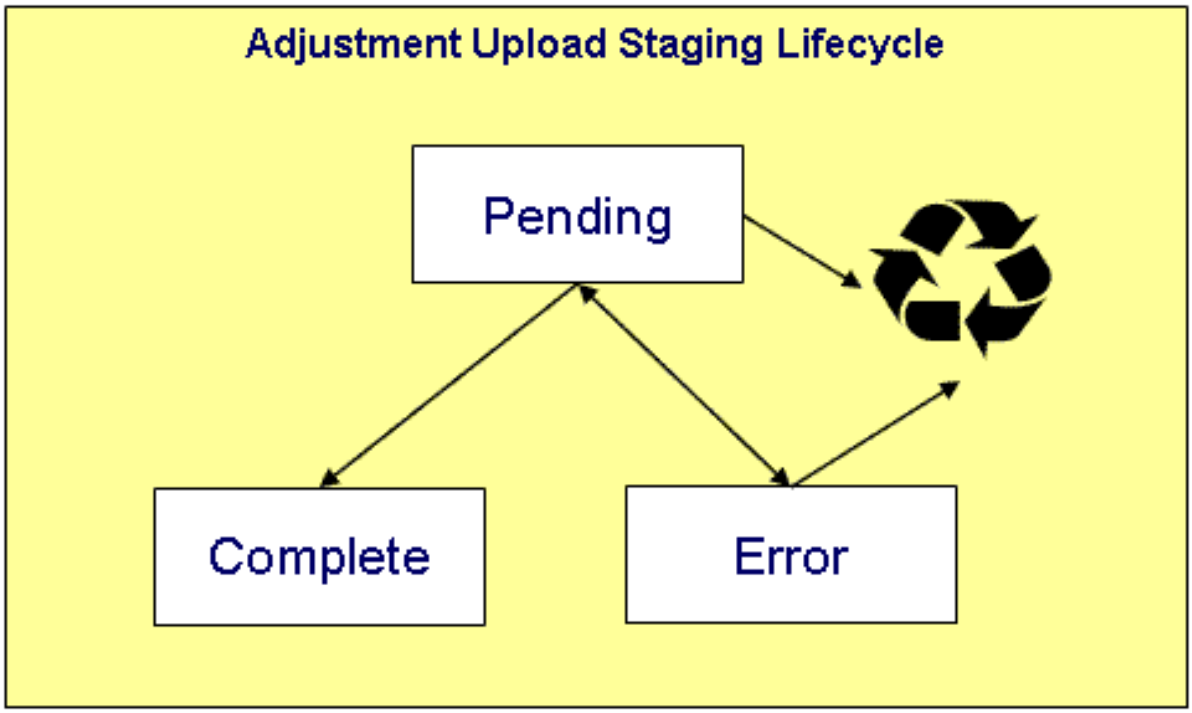


- **Pending.** An adjustment staging control record is created in this state. The **C1-ADUP1** process selects **pending** adjustment staging control records for validation.
- **In Progress.** The **C1-ADUP1** process sets a **pending** record to **in progress** when the totals on the adjustment staging control are successfully validated against the totals from the associated adjustment upload staging records.
- **Complete**. The **C1-ADUP2** process sets the adjustment staging control's status to **complete** when all adjustment upload staging records linked to the adjustment staging control are **complete**.
- **Error.** The **C1-ADUP1** process sets a **pending** record to **error** if the adjustment staging control fails validation. The status may be set back to **pending** after the error is fixed.

- **Held.** The **held** status is available for situations where you want to prevent or delay the upload of a batch of adjustment staging records. The status may be set back to **pending** when the batch of records is ready for upload.
- **Pending, error** and **held** records may be deleted.

The Lifecycle of an Adjustment Upload Staging Record

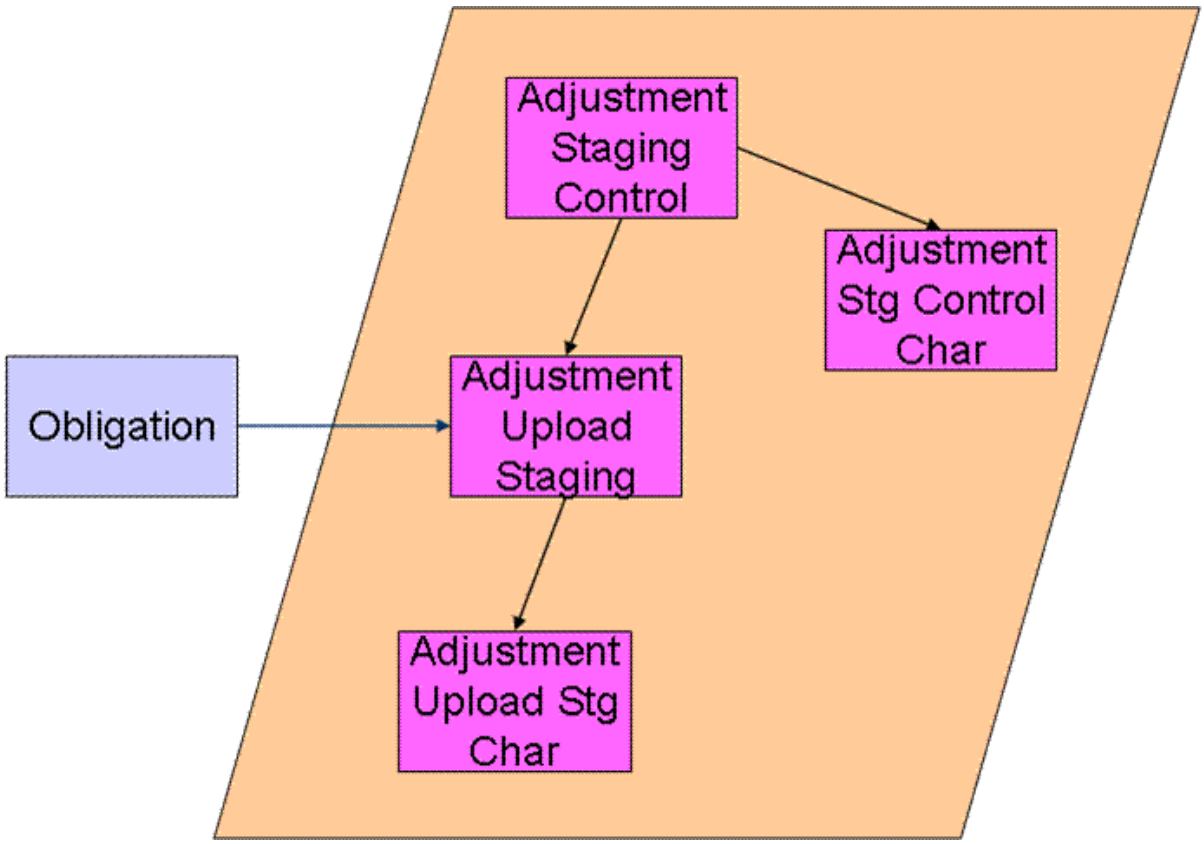
The following diagram shows the possible lifecycle of an adjustment upload staging record.



- **Pending.** Adjustment upload records are created in this state. The **C1-ADUP2** process selects **pending** adjustment upload records and creates adjustments for each of them.
- **Complete.** The **C1-ADUP2** process sets a **pending** record to **complete** if an adjustment is successfully created.
- **Error.** Either **C1-ADUP1** or **C1-ADUP2** process sets **pending** record to **error** when it encounters an error during the upload process. The status may be set back to **pending** after the error is fixed.
- **Pending** and **error** records may be deleted.

Process C1-ADUP1 - Preprocess Adjustment Uploads

The batch process identified by batch process ID **C1-ADUP1** refers to the background process that validates adjustment staging control records and populates obligation IDs on adjustment upload staging records that do not specify an obligation ID.



Phase 1 - Validate Adjustment Staging Controls

The following points describe, at a high level, the first phase of the adjustment upload pre-process:

- For each **Pending** adjustment staging control,
- Check that the record count on the adjustment staging control record equals the number of adjustment upload staging records that are linked to the adjustment staging control.
- Check that the total adjustment amount on the adjustment staging control record equals the sum of the adjustment amounts from the adjustment upload staging records that are linked to the adjustment staging control.
- If the adjustment staging control passes validation, set its status to **In Progress**. Otherwise, set the status to **Error**. Create a To Do entry using the inputs To Do type and To Do role (if supplied) for adjustment staging control errors. (Complete any outstanding To Do entries for the adjustment staging control before creating a new To Do entry.)
- If no errors occur, perform To Do cleanup by completing any outstanding To Do entries that were previously created for the adjustment staging control.

Note:

You can fix errors by going to the To Do entry and drilling into the adjustment staging control page. Don't forget to change the adjustment staging control's status back to **Pending** after fixing the error.

Phase 2 - Populate Obligation ID

The following points describe, at a high level, the second phase of the adjustment upload pre-process:

- For each **Pending** adjustment upload staging that is linked to an **In Progress** adjustment staging control AND does not have the obligation ID,
- Execute the **Determine Obligation** algorithm that is plugged in on the *adjustment type*.

- If the algorithm returns a valid obligation ID, stamp that obligation ID onto the adjustment upload staging. If the algorithm also returns an indication that the adjustment is to be put into suspense, set the suspense flag on the adjustment upload staging to **In Suspense**. Refer to *Suspense Adjustments* for more information on how suspense adjustments are handled.
- If the algorithm returns an error, set the adjustment upload staging record's status to **Error**. Create a To Do entry using the inputs To Do type and To Do role (if supplied) for adjustment upload staging errors. (Complete any outstanding To Do entries for the adjustment upload staging before creating a new To Do entry.)
- If no errors occur, perform To Do cleanup by completing any outstanding To Do entries that were previously created for the adjustment upload staging.

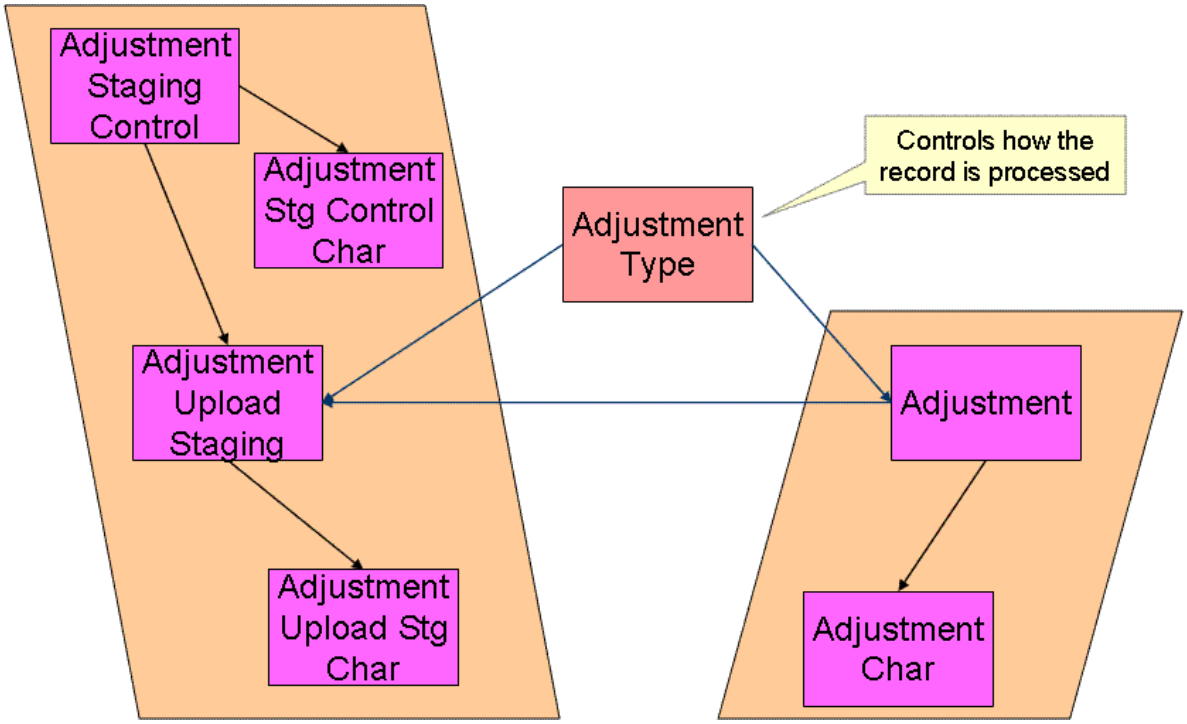
Note:

You can fix errors by going to the To Do entry and drilling into the adjustment upload staging page. Don't forget to change the adjustment upload staging's status back to **Pending** after fixing the error.

Process C1-ADUP2 - Upload Adjustments

The batch process identified by batch process ID **C1-ADUP2** refers to the background process that creates adjustments for all adjustment upload staging records that are stamped with an obligation ID.

The following diagram and section describe, at a high level, the processing done in the **C1-ADUP2** background process.



- For each **Pending** adjustment upload staging that has an obligation ID,
- Add a frozen adjustment using the amount, creation date and adjustment type on the staging record.

Note:

If the type of adjustment being uploaded is one that is calculated, the algorithm that's plugged in on the adjustment type will calculate the adjustment amount and generate associated calculation lines (if applicable). See *Calculated Adjustments* for more information.

- If adjustment creation is successful, update the adjustment ID on the staging record and set the staging record's status to **Complete**.

- If adjustment creation results in an error, set the staging record's status to **Error**. Create a To Do entry using the To Do type on which this batch process (**C1-ADUP2**) is defined as creation process. (Complete any outstanding To Do entries for the adjustment upload staging before creating a new To Do entry.)
- If no errors occur, perform To Do cleanup by completing any outstanding To Do entries that were previously created for the adjustment upload staging.

Suspense Adjustments

What Are Suspense Adjustments?

When the adjustment upload preprocessor is unable to identify a valid obligation ID, it can post the adjustment to a suspense obligation. This is similar to how a payment is posted to a suspense account when a valid account ID could not be identified during payment upload.

Putting adjustments in suspense is optional. Therefore, this logic sits in a plug-in spot. If suspense adjustments are applicable to you, you must plug in your suspense logic in a **Determine obligation** algorithm on *Adjustment Type*.

How Are Suspense Adjustments Resolved?

A background process can be run periodically to automatically resolve suspense adjustments. Since rules for resolving suspense may vary, this logic sits in a plug-in spot. You must plug in your resolve logic in a **Resolve Suspense** algorithm on *Adjustment Type*.

Process C1-ADURS - Resolve Suspense Adjustments

The batch process identified by batch process ID **C1-ADURS** refers to the background process that resolves suspense adjustments.

For each adjustment upload staging that is **In Suspense**, execute the **Resolve Suspense** algorithm that is plugged in on the *adjustment type*.

Managing Payment Setup

The following topics describe maintenance transactions related to payment topics.

Fastpath:

Refer to *Tender Management and Workstation Cashiering* before setting up the control tables described in this section.

Setting Up Payment Segment Types

Every obligation references an obligation type. Among other things, the obligation 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)*

Where Used

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

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 taxpayer'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.

External ID Required	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.

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 taxpayers 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 Obligation*
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 **Obligation** to define the obligation whose account will hold uploaded payments with an invalid account. Refer to *Payment Upload Error Obligations* for more information about suspense obligations. Also note, because the payment upload process simply books payments that reference invalid accounts to the account associated with this obligation, this account should belong to an account type with the appropriate payment distribution algorithms. This may entail creating a new account type that will only be used on these "suspense accounts". This account type would need the following algorithms:

- We'd recommend using a simple payment distribution algorithm like *PYDIST-PPRTY* (distribute payment based on obligation type's payment priority).
- We'd recommend using an overpayment distribution algorithm like *OVRPY-PPRTY* (distribute overpayment to highest priority obligation 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*.

Setting Up Payment Cancellation Reasons

Open **Admin Menu** > **Pay Cancel Reason** to define your payment cancellation reason codes.

Description of Page

Enter an easily recognizable **Cancel Reason** and **Description** for the payment cancellation reason.

Turn on the **NSF Charge** switch if the system should invoke the non-sufficient funds (NSF) algorithm when a tender is cancelled using this reason code. Refer to *NSF Cancellations* for more information.

The following fields are used to change an account's compliance rating if a tender is canceled using the respective reason code.

- Use **Affect Compliance Rating By** to define how tenders canceled using this reason will affect the account's compliance rating. This should be a negative number. A taxpayer's compliance rating is equal to the start compliance rating amount defined on the installation record plus the sum of compliance rating demerits that are currently in effect.
- Use **Months Affecting Compliance Rating** to define the length of time the demerit remains in effect. This information is used to define the effective period of the compliance rating demerit record.

Fastpath:

For more information, refer to *Account - Collections*.

Note:

The payor gets the compliance rating hit. When you cancel a tender you must specify a cancellation reason. If the cancellation reason indicates a compliance rating demerit should be generated, the system levies the compliance 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.

Automatic Payment Options

If your taxpayers 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.

Fastpath:

Refer to *Automatic Payments* for more information about how automatic payments are handled in the system.

Setting Up Auto-Pay Route Types

Auto Pay Route Types are used to control when and how automatic payment requests are routed to a financial institution, and when the general ledger is impacted. Select **Admin Menu** > **Auto Pay Route Type** to define your route types.

Description of Page

To modify an auto pay route type, simply move to a field and change its value.

To add a new route type, press + to insert a row, then fill in the information for each field. The following fields display:

Route Type	The unique identifier of the route type.
Description	The description of the route type.
Tender Source	The background process that routes automatic payment requests to a financial institution (e.g., the automated clearing house interface) will mark each automatic payment's associated tender with a tender control for audit and control purposes. The following points describe how this happens:

- When the system sees that it's time to send an automatic payment to a financial institution, it looks at the automatic payment's auto pay source.
- Every auto pay source references an auto pay route type.
- Every auto pay route type references a tender source.
- A **Tender Source** has a tender control for each group of tenders deposited / interfaced together one batch.
- The system marks each automatic payment's associated tender with the latest tender control for the **Tender Source**. The system will create a new tender control each time it routes automatic payments to the tender source.

Refer to *Managing Payments Interfaced From External Sources* for more information about tender source and tender control.

Extract Batch Code

This field defines the background process that interfaces the automatic payment requests to the financial institution.

Auto Pay Date Calculation Alg

This algorithm populates 3 dates associated with the automatic payment: 1) the date the automatic payment will be sent to the financial institution, 2) the date the general ledger will be impacted by the automatic payment, 3) the date of the payment.

If you haven't done so already, you must set up this algorithm in the system. To do this, create a new algorithm (refer to *Setting Up Algorithms*). On this algorithm, reference an Algorithm Type that populates automatic payment dates. Click *here* to see the algorithm types available for this plug-in spot.

Where Used

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

Setting Up Auto-Pay Source Codes

A unique **Auto Pay Source** must exist for every bank / credit card company / bill payment service that your taxpayer'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 **Auto Pay Route Type** controls when and how automatic payment requests get routed to a financial institution. It also controls when the general ledger is impacted by the automatic payments financial transaction. Refer to *Setting Up Auto-Pay Route Types* for more information.

The **Work Calendar** defines the financial institution's workdays. This information is used to determine the date on which automatic payment requests will be sent to the financial institution. Refer to *Setting Up External Workday Calendars* for more information.

The **Validation Algorithm** defines how the system validates the taxpayer'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 taxpayer'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*.

Managing Bill Setup

The following topics describe maintenance transactions related to billing topics.

Defining Bill Segment Types

Every obligation references an obligation type. Amongst other things, the obligation 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.

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.

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 obligation type and the bill segment algorithm used on an obligation type.

For each bill segment type, define the **Financial Algorithm**. The logic embedded in this algorithm constructs the financial transaction associated with the 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 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*.

Define the **Pre-creation Algorithm**. The logic embedded in this algorithm retrieves detailed information needed to bill the 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 retrieves details needed for billing in the appropriate manner. 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*.

Setting Up Billable Charge Templates

A user creates a billable charge whenever a taxpayer should be levied an ad hoc charge. For example, if a taxpayer requires a review of their property assessment, you may charge them an administration fee.

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. Refer to *Uploading Billable Charges* for more information.

A billable charge must reference an obligation. This obligation behaves just like any other obligation:
- **Bill segments are created for the obligation.** Whenever billing is performed for an account with billable charge obligations, the system creates a bill segment for each unbilled billable charge.
- **Payments are distributed to the obligation.** Payments made by an account are distributed to its billable charge obligations just like any other obligation.
- **Overdue debt is monitored.** The credit and collections process monitors billable charge obligations 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 an obligation 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.

Fastpath:

Refer to *How To Create An Ad-hoc Bill* 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 taxpayer. 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.

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 obligation types that can use each template. Obviously, only **billable charge** obligation types (as defined on the obligation type's special role) will reference billable charge templates.

Billable Charge Template - Main

Open **Admin Menu** > **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 taxpayer'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 taxpayer'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 taxpayer).
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 An Ad-hoc Bill*.

Billable Charge Template - Line Characteristics

Open **Admin Menu** > **Billable Charge Template** and navigate to the **Line Characteristics** tab 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.

The following fields display:

Characteristic Type	The type of characteristic.
Characteristic Value	The value of the characteristic.

Billable Charge Template - RQ Details

Open **Admin Menu** > **Billable Charge Template** and navigate to the **RQ Details** tab to define your billable charge templates service quantities.

Description of Page

The following fields display:

Sequence	Specify the sequence number of the RQ.
UOM	Select the unit of measure of this RQ. One or more of UOM, TOU, or RQ identifier must be selected.
TOU	Select the time of use period.
RQ Identifier	Select the RQ identifier.
Rate Quantity	Specify the number of units of this rate quantity.

Where Used

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

Bill Charge Line Type

Note:

Background information. Before using this page, you should be comfortable with the topics described under *Setting Up Billable Charge Templates* and *Uploading Billable Charges*.

Billable charge line types will simplify the effort required to interface billable charges from an external system. Each line type contains values that will be defaulted onto the line details associated with the uploaded billable charges. Obviously, this defaulting is possible only if you specify a billable charge line type on the billable charge upload staging lines.

To set up billable charge line types, select **Admin Menu** > **Bill Charge Line Type** .

Description of Page

Enter an easily recognizable **Bill Charge Line External Type** and **Description**.

Use **Currency Code** to define the currency to be defaulted onto billable charge upload lines that reference this line type.

Use **Show on Bill** to define the value to be defaulted into the Show on Bill indicator on billable charge upload lines that reference this line type.

Use **App in Summary** to define the value to be defaulted into the App in Summary indicator on billable charge upload lines that reference this line type.

Use **Memo Only, No GL** to define the value to be defaulted into the Memo Only, No GL indicator on billable charge upload lines that reference this line type.

Use **Distribution Code** to define the values to be defaulted into the Distribution Code field on billable charge upload lines that reference this line type.

Where Used

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

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 **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 value is **Mass Cancel**. 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.

Defining Bill Cycles and Bill Periods

This section discusses issues related to defining bill cycles and bill periods in the system.

An account may reference a bill cycle. An account's bill cycle may be used to control cyclical billing.

In this section, we describe how to design and set up this cycle. In addition, we discuss how to set up bill period schedules. These are used to define the bill segment end date for special types of obligations.

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:

- An account can reference a bill cycle. The bill cycle's schedule controls when a cyclical billing process 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, your implementation may use a concept of "window billing" where the system attempts to produce bills for accounts over a few nights. This concept is useful if your billing is based on information being sent from an external source. It 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 successfully created, it is completed immediately and ready to send to the taxpayer. If a bill cannot be created, the system will create a bill in the "error" state and initiate a To Do entry with a message that can be analyzed by your billing staff. 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.

Designing Your Bill Cycles

The number of bill cycles is determined by how much you want to stagger the billing of your taxpayers. You may do this to smooth out calls you may receive for your call center with questions about tax bills.

How Does An Account Get Its Bill Cycle?

Most accounts are created behind-the-scenes through taxpayer registration processing. An account created like this doesn't have a bill cycle. Rather, it sits in limbo awaiting the activation of its first obligation that should be billed using cyclical billing. When an obligation is activated, an activation algorithm should assign the account to an appropriate cycle based on business rules.

Note, an account may be configured to protect its bill cycle from being changed by an algorithm. 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 obligations.

Protecting An Account's Bill Cycle

An account has a single bill cycle, but may have multiple obligations that are billed via cyclical billing process. In this case, when a cyclical obligation is activated, you may not want an algorithm to change the account's bill cycle. To prevent this you need to turn on the account's **protect bill cycle** flag.

When the last obligation for an account is stopped, the protect bill cycle flag is reset. This is to ensure that if the taxpayer starts a new obligation in the future the bill cycle can be set based on the new obligation's activation algorithm rules.

Setting Up Bill Cycles

An account may reference a bill cycle. The bill cycle defines the schedule for cyclical billing of its accounts. 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 a cyclical billing process should 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 taxpayers to receive bills. When the cyclical billing process next runs, it should delete all unfrozen bills and recreate 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*.

Setting Up Bill Periods

Some obligation types reference a bill period. The bill period defines when the obligation's bill segments are produced and the respective end date of each bill segment.

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 obligations 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 obligations 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 obligations that use a bill period.

Other Financial Transaction Topics

Various topics about financial transactions are discussed in this section.

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 taxpayer instead of initiating electronic funds transfer directly to the taxpayer's bank.

What Is A Payment Advice?

Payment advice is a money order that is established at the initiative of the tax authority. When a bill is completed, the tax authority sends the taxpayer a document that indicates a payment amount and the taxpayer's bank details. If the taxpayer agrees to the information on the payment advice, he or 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 taxpayer'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 taxpayer. The taxpayer sends the approved payment advice directly to the clearinghouse.

The system does not provide sample processes for extracting and printing payment advice information. Your implementation team would have to create these.

Setting Up The System To Enable Payment Advices

You must set up a *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.

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.

The base package provides a sample Distribution Rule - Create Payment *C1-PAYFRM* that caters for multiple distributions rule values.

Common Distribution Rules

Tax authorities have complex payment distribution rules. Most of these rules follow a similar pattern, with a few key differences.

The following lists common payment distribution methods:

- Pay a specific obligation.
- Pay all the accounts of the taxpayer.
- Pay an account.
- Pay a specific filing period. This is common for estimated payments where a payment form indicates the filing period. Some tax authorities may use paper forms in check processing that indicate the filing period to which the payment should apply. This information is keypunched with the payment. The base package provides Distribution

Rule - Create Payment *C1-PAYFRM* for paying a filing period. Refer to *Directing a Payment to a Form or Filing Period* for more details.

- Pay a tax form. The payment is directed to the obligation on the tax form. The base package provides Distribution Rule - Create Payment *C1-PAYFRM* for paying a form. Refer to *Directing a Payment to a Form or Filing Period* for more details.
- Pay a pay plan. The payment is directed to the obligations that the pay plan covers. The base package provides a Distribution Rule - Create Payment algorithm *C1-DR-PAYPP* for paying a pay plan. Refer to *Distributing Payments for Pay Plans* for more details.
- Pay a collection case. The payment is directed to what the collection case is collecting on - e.g. obligations, assessments, etc.
- Pay a collection letter. A collection letter can be generated from overdue processing. The payment is directed to what the overdue process is collecting on - e.g. obligations, assessments, etc.

Payments Linked To Assessments

When dynamic credit allocation is practiced, a payment targeted to a specific assessment must have the payment FT referencing the assessment's FT ID. If there is excess credit on the payment segment, it is dynamically allocated to any other assessments for the obligation.

The base package provides a payment freeze algorithm *C1-PYFZ-PYAS* that links a payment FT to a specific assessment. This algorithm looks for a match value on the payment that references the assessment. The assessment is linked to the payment by populating the assessment's FT ID on the payment FT's Group FT ID field.

For more information, refer to *Credit Allocation*.

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, **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 obligation, you'd reference a characteristic that its value identifies an obligation. 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 Menu** > **Distribution Rule** and navigate to the **Algorithms** tab to set up the algorithms appropriate for your distribution rule.

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 a *Feature Configuration* to define parameters that control various payment event distribution options. The following is an example of how the Feature Configuration would look for an implementation:

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.

Set Up Account Type

If you do dynamic credit allocation, set up the payment freeze algorithm *C1-PYFZ-PYAS*.

We recommend specifying **payment distribution** and **overpayment** algorithms because they are used as default logic in case distribution rules are not supplied.

Set Up Match Type

If you use set the payment freeze algorithm *C1-PYFZ-PYAS*, set up a match type for assessment. This match type should NOT reference an override payment distribution algorithm (if this algorithm is blank, the account type's payment distribution algorithm is used). Refer to *Payments Linked to Assessments* for more details.

Payables Cash Accounting

In some cases certain amounts such as sales tax distributions and 3rd party liabilities are not truly payable until the taxpayer 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).

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

Accrual accounting means that all payables are booked when the debt financial transaction (bill segment or adjustment) is created.

If the payable is subject to payables cash accounting, the liability is not booked when the bill segment or adjustment is created, rather, the amount of the liability is placed in a "holding" GL account. When the taxpayer pays, the moneys are transferred from the "holding" GL account to the true tax payable account.

The following is an example of the financial events that transpire when a simple sales tax assessment is posted and a payment is received using accrual accounting.

Event	***GL Accounting***	***County A Payable Balance***
Tax assessment created	A/R 110 Revenue - State <100> Payable - County A <10>	(10)
Payment received	Cash 110 A/R <110>	(10)

In the above example, you'll notice that the payable is booked when the adjustment is created. Let's contrast this with what takes place if the payable is subject to payables cash accounting.

Event	***GL Accounting***	***County A Payable Balance***	***County A Holding Balance***
Tax assessment created	A/R 110 Revenue - State <100> Holding - County A <10>	0	(10)
Payment received	Cash 110 A/R <110> Holding - County A 10 Payable - County A <10>	(10)	0

Notice that when the adjustment is created, the payable for the county is not booked, rather, the amount is placed in a "holding" GL account. When the taxpayer pays, the moneys are transferred from the "holding" GL account to the true **County A** payable account.

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 amounts subject to cash accounting rules?
- 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 taxpayer overpays?
- What happens if the obligation is subject to penalty and interest and dynamic credit allocation?

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 amount subject to cash accounting rules. For example, for sales tax distribution, a pair of payable and holding accounts is required for each separate distribution amount.

Without unique distribution codes for each payable and holding account, the system cannot keep track of how much of a given amount is being held, awaiting payment.

Cash Accounting and Incurring Debt

When calculating debt that includes payables, the distribution codes associated with the payables (for example on the rate components that calculate payables) must be the holding payable distribution codes. No other logic is needed for cash accounting at this time.

Cash Accounting and Relieving Debt

The following section describes functionality required for obligations that practice cash accounting and do NOT practice penalty and interest or dynamic credit allocation.

Caution:

Dynamic Credit Allocation. Refer to *P&I and Cash Accounting* for functionality required for obligations that do practice penalty and interest or dynamic credit allocation.

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 taxpayer.

Event	*GL Accounting*
Adjustment is created	A/R 110

	Revenue - State <100> County 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	County Holding 10 County Payable <10>
Net affect of the above	Cash 110 A/R <110> County Holding 10 County 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 obligation. 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 County holding amount being transferred to the County payable account. The entire holding amount was transferred because the obligation 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***	***County Payable Balance***	***County Holding Balance***
Adjustment created	A/R 110 Revenue - State <100> County Holding <10>	0	(10)
Partial payment received	Cash 27.50 A/R <27.50> County Holding 2.50 County 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), for example an overpayment transferred one obligation 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 "negative 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 taxpayer has overpaid their taxes.

Note:

This is an example of a tax that is "billed" where one obligation is used for ongoing billing of the tax amount.

Event	*GL Accounting*	*County Payable Balance*	*County Holding Balance*
Bill segment created	A/R 110 Revenue - State <100> County Holding <10>	0	(10)
Payment received	Cash 110 A/R <110> County Holding 10 County Payable <10>	(10)	0
Cancel	A/R <110> Revenue 100 County Holding 10	(10)	10
Rebill	A/R 27.50 Revenue <25> County Holding <2.50>	(10)	7.50

You'll notice that the amount payable to the county still indicates $10 (the amount of amount that was paid by the taxpayer). However, you'll notice that the county holding balance is 7.50 (debit). This looks a bit odd, but it's correct. Remember that at this point, the taxpayer 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*	*County Payable Balance*	*County Holding Balance*
		(10)	7.50

Bill segment created	A/R 55 Revenue <50> County Holding <5>	(10)	2.50
Bill segment created	A/R 110 Revenue <100> County Holding <10>	(10)	(7.50)

In the unlikely event of a payment being received while the county 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 taxpayer, 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***	***County Payable Balance***	***County Holding Balance***	***Obligation's Payoff Balance***
Bill segment created	A/R 110 Revenue <100> County Holding <10>	0	(10)	110
Payment received	Cash 110 A/R <110> County Holding 10 County Payable <10>	(10)	0	0
Cancel	A/R <110> Revenue 100 County Holding 10	(10)	10	(110)
Rebill	A/R 27.50 Revenue <25> County Holding <2.50>	(10)	7.50	(82.50)
Payment refunded (via an A/P adjustment)	Cash <82.50> A/R 82.50 County Holding <7.50> County Payable 7.50	(2.50)	0	0

We understand this is tricky, but consider this - when a cash accounting taxpayer makes a payment, the system transfers county holding credit balances to county 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 taxpayer, the system should transfer county holding debit balances to county 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 taxpayer overpays a tax amount (i.e., we receive more cash than receivables), we strongly recommend you set up the system to NOT keep the excess credit on the taxpayer's regular obligation. Rather, we recommend you segregate the receivable onto an "excess credit" obligation. If you do this, the system can transfer any excess credits to the regular obligations at bill completion time or at form posting 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 obligation and the normal obligation had $10 of held payables.

Fastpath:

Refer to *Overpayment Obligations* for how to set up the system to segregate overpayments on a separate obligation.

Note:

Why not keep excess credits on a taxpayer's regular obligation? 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 obligation, the system knows to treat any transference of these credits as "payments" and therefore it can transfer holding balances to true payables.

Event	***Normal Obligation GL Accounting***	***Excess Credit Obligation GL Accounting***
Bill segment created	A/R 110 Revenue <100> County Holding <10>	
Payment of $300 is received	Cash 110 A/R <110> County Holding 10 County Payable <10>	Cash 190 Overpay <190>
Bill segment created	A/R 110 Revenue <100> County Holding <10>	
Transfer excess credit amount to normal obligation (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 county holding is relieved in proportion to the amount of receivables that are reduced by the transfer	County Holding 10 County Payable <10>	

Net affect of the transfer	Xfer 110 A/R <110> County Holding 10 County Payable <10>	Overpay 110 Xfer <110>

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 *account types*; 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 taxpayer's oldest debt.

Accounting Method Defined On Your Account Types

You define the type of accounting method that is practiced (*balance-forward versus open-item*) on your *account types*. The account type should be configured based on the accounting method practiced for that tax type.

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 taxpayers, the system matches credit financial transactions (FT's) to debit FT's under a *match event*. The following are important points about match events.

- A match event may contain an unlimited number of FT's. For example, the match event can match 2 credit FTs against a single debit FT.
- A match event contains FTs associated with a single account. While the FTs under a match event may belong to multiple obligations, all FTs under a match event must belong to the same account.
- The status of the match event is **balanced** when the sum of the debits equals the sum of the credits. If debits do not equal credits, the status of the match event is **open** and the various FTs would still affect the taxpayer's aged debt. Refer to *Match Event Lifecycle* for more information.

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 an account type 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 FTs with the debit and credit FTs from bill segments and adjustments. The FTs that are linked to the match event are controlled by the payment's **match type** and **match value** (payments made by open-item taxpayers 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 taxpayers that pay automatically (i.e., direct debit taxpayers). 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.

Fastpath:

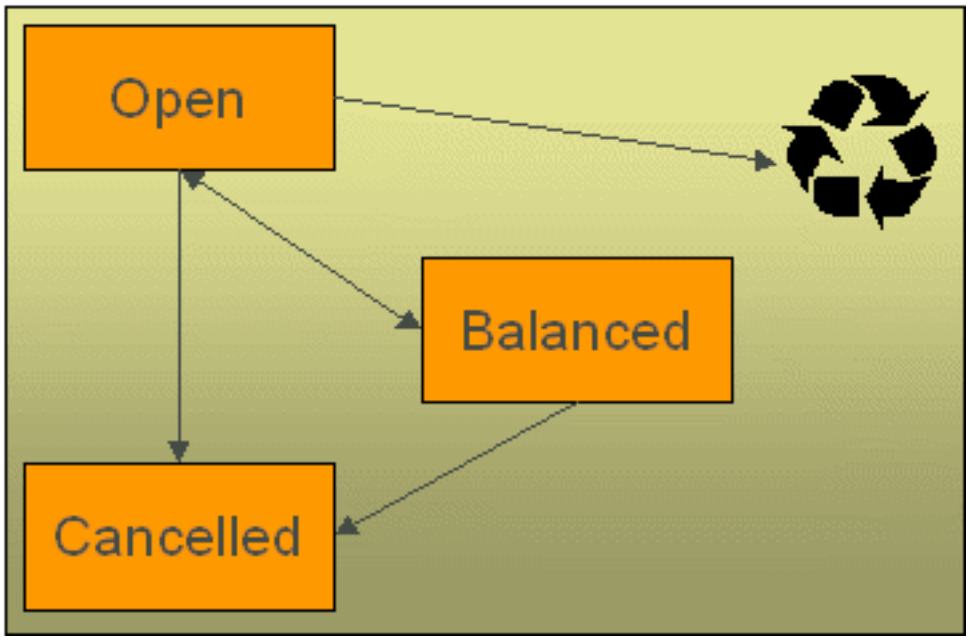
Refer to *Bill Lifecycle* for more information about what happens during bill completion.

- The system creates a match event when an obligation closes and the obligation has unmatched FTs. For example, consider an obligation for debt that is written off. This obligation closes when the system creates transfer adjustments to transfer the debt to a write-off obligation (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.
- 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 taxpayer 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 obligation on the match event. It's worth stressing that a match event may contain FT's from many obligation's and each obligation'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 account type.
- 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 account type 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 obligations 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 To Do entry to highlight the existence of open match events.

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 account types).

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 taxpayers. 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, 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 taxpayers may dispute FT's that they are not comfortable paying. For example, a taxpayer who receives a bill with an anomalous charge may decide to dispute it.

When an open-item taxpayer disputes a charge, a user creates a match event and links the disputed FT(s) to it. This match event will be in the **open** state (because it does not contain FT's that sum to zero). In addition, the match event's "disputed switch" is turned on.

Note:

Alerts. An alert is displayed on control central to highlight the existence of disputed match events (if the appropriate alert algorithm is plugged in). In addition, you can also set up a To Do entry to highlight the existence of disputed match events.

While the dispute is being researched, the disputed amount will not affect aged debt, but it still forms part of the taxpayer's balance.

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).

If the dispute goes in the taxpayer's favor,

- 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.

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 taxpayer's current debt is satisfied.
- The amount of the overpayment will be kept on a separate obligation (this only happens if you plug-in the appropriate Overpayment Distribution algorithm on your account types). Refer to *Overpayment Obligations* 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 obligation) will not be matched.
- When future bills are completed, the credit balance on this "overpayment obligation" will be transferred to the "real obligation's" when future bills are completed (if you have plugged in the appropriate bill completion algorithm on the overpayment obligation's obligation 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 obligation. 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 such as overpayments.
- At some point in the future, the overpayment will be exhausted (i.e., all funds will be transferred to "real obligations"). At that point in time, the overpayment obligation will close (assuming you set up the overpayment obligation's obligation 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 obligation's". Refer to *When Are Match Events Created* for information about how the system creates match events when an obligation 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.
- **Obligation ID.** This match type should reference an override payment distribution algorithm that distributes the payment based on the obligation ID specified on the payment (in match value). Refer to *Payments And Match Events* for more information.
- **Pay Plan.** This match type should NOT reference an override payment distribution algorithm (if this algorithm is blank, the account type's payment distribution algorithm is used). Refer to *Pay Plans* for more information.

Match Event Cancellation Reason Setup

The number of match event cancellation reasons that you will need is dependent on the number of ways your organization can justify the cancellation of a match event. At a minimum, you will probably need the following match event cancellation reasons:

- **FT Cancellation.** This cancel reason should be referenced on the account type 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.

Account Type Setup

The following points describe *account type* 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" obligation. Refer to *Overpayments* for more information.

Overpayment Obligation Type Setup

Specify a **bill completion** algorithm that causes the credit amount on overpayment obligation's to be transferred to newly create debt (created when the bill is created). This algorithm transfers an overpayment obligation's balance to regular obligation'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 To Do types are supplied with the base package:

- **TD-MODTL**. This To Do type highlights the presence of open, disputed match events.
- **TD-MONTL**. This To Do type highlights the presence of open, non-disputed match events.

Each of the above To Do types should be configured with the roles that work on entries of each type.

In addition, the account management group and/or divisions from which the default roles are extracted should be updated to define the role that should be defaulted for each of the above To Do types.

Fastpath:

Refer to *The Big Picture Of To Do Lists* for more information about To Do lists.

Setting Up Match Types

Most payments are distributed amongst obligations using the payment distribution algorithm specified on the payment's account's account type. This algorithm decides how to distribute a payment amongst an account's existing debt if the taxpayer doesn't specify how the payment should be distributed.

A taxpayer can specify how a payment is distributed by specifying a match type and match value on their payments. Consider the following examples:

- Taxpayers that are subject to open-item accounting (this is defined on the account's account type) tell the system exactly which debt is covered by their payments. For example, an open-item taxpayer might make a payment in respect of bill ID **123919101919**.
- Even non open-item taxpayers can direct payments to specific obligations. For example, the system allows a balance-forward taxpayer's payment to be directed to a specific obligation (however, they cannot direct payments to specific bills as only open-item taxpayers 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 account type.

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

Regulations or other restrictions may require some organizations to account for the finances of each of its departments as a separate entity.

To track the finances of departments separately, the organization sets up a fund for each department. A fund is an accounting entity with its own self-balancing set of accounts. Each fund has its own "sub general ledger" with its own

chart of accounts, and within each fund, its debits equal its credits at all times. This allows the organization 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.

A single obligation may have debt related to different funds. For example, tax liability and penalty and interest may go to the same "revenue" fund, and a collection fee may go to a "collection" fund.

In fund accounting, debits and credits must balance for the whole general ledger and debits and credits within each fund must balance.

Fund Accounting Example

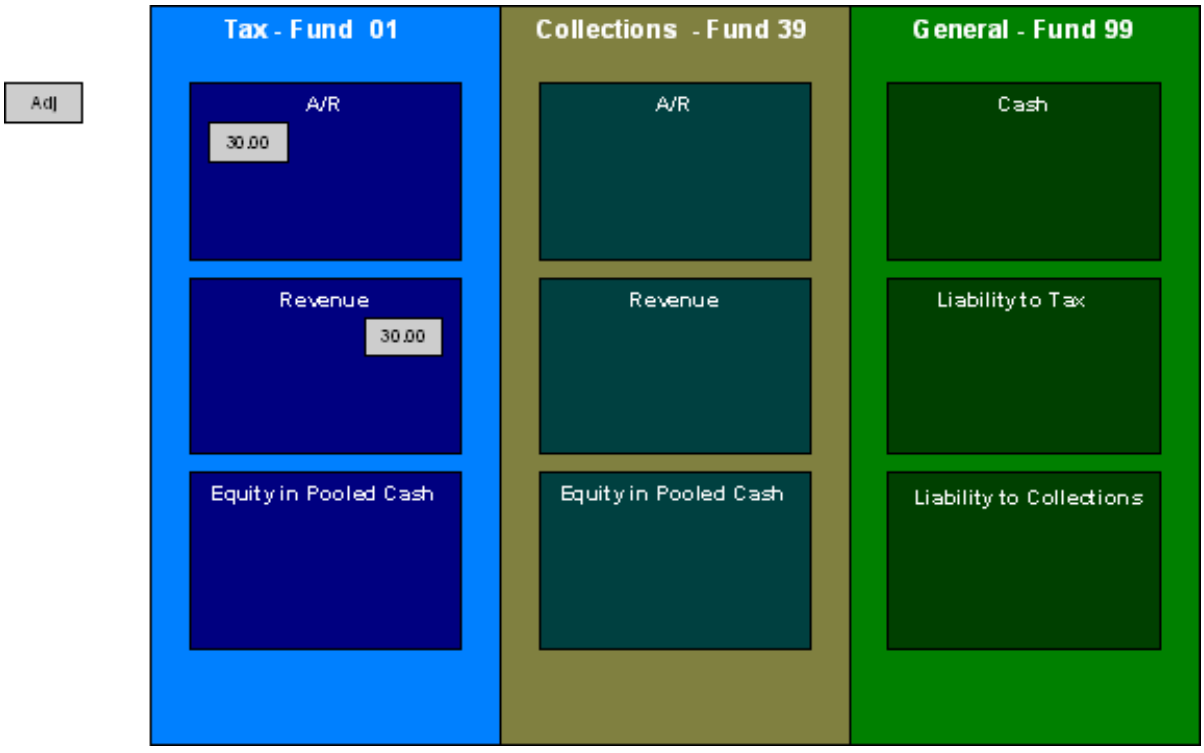
Consider an obligation that owes tax liability and then incurs a collection fee. Note that penalty and interest would also accrue, but are not shown in this example.

The accounts receivable (A/R) distribution code to use for financial transactions that affect A/R is linked to the obligation type. The assumption is that the fund for this distribution code is the same "revenue" fund.

For this example, three funds exist:

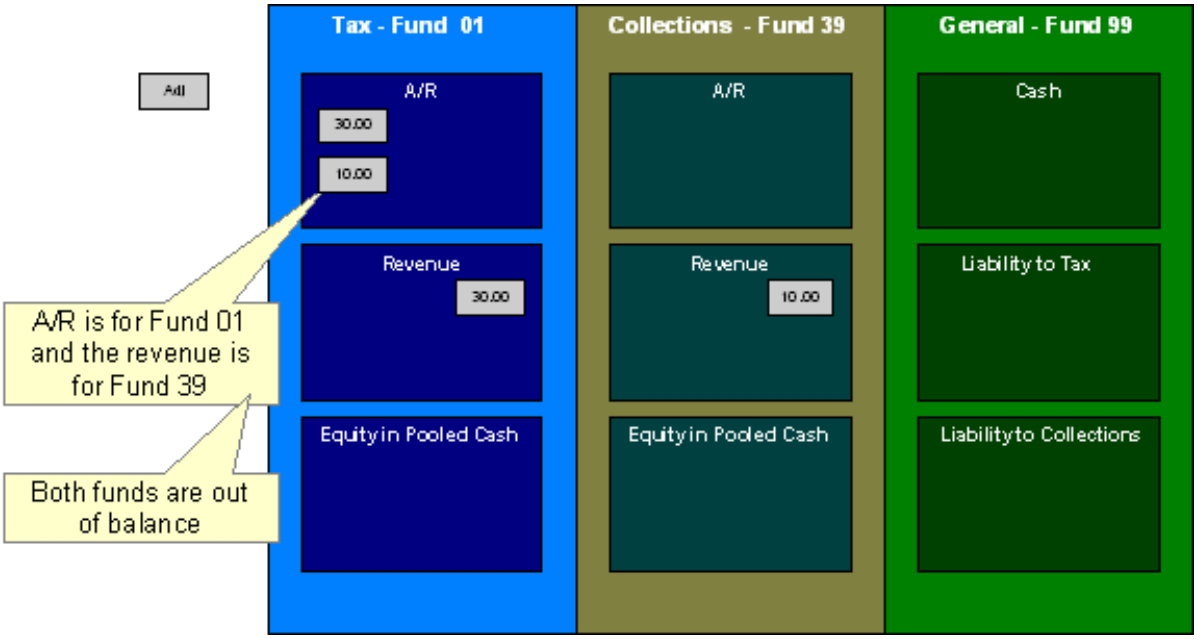
- Tax (fund 01)
- Collections (fund 39)
- General (fund 99)

When a tax is levied, the following adjustments are made to the tax fund account producing the following GL entries.



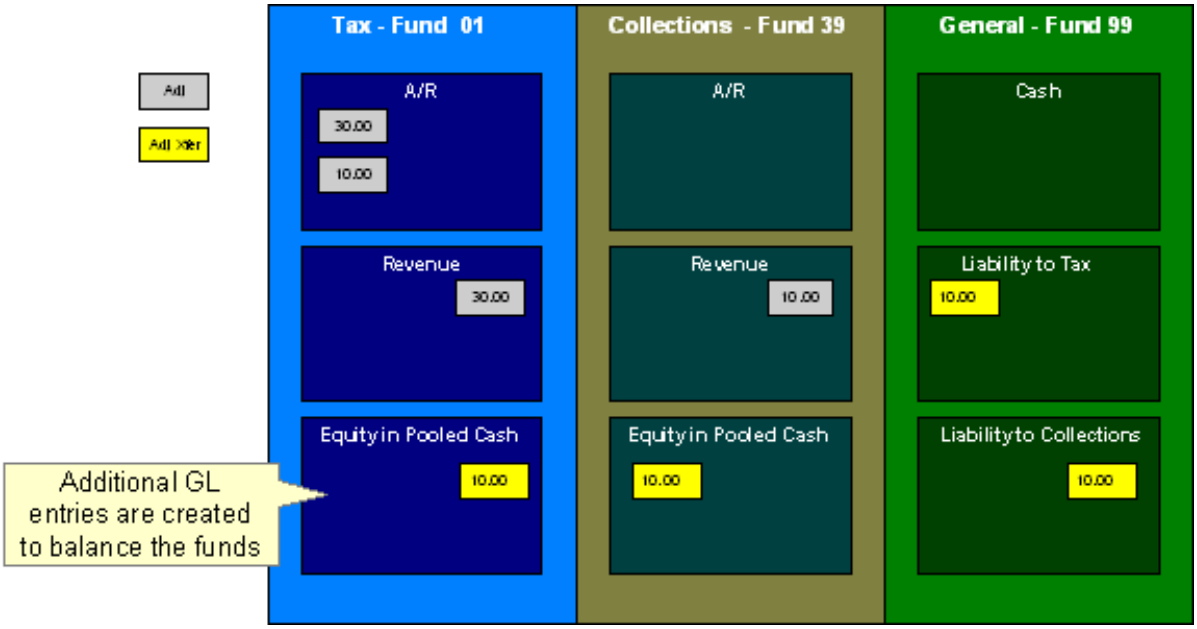
For the tax fund, the GL details of the bill include a debit to the accounts receivable (A/R) account and credits to the revenue account. The taxing authority is owed the entire portion of the adjustment by the taxpayer. The tax fund is balanced.

The following diagram illustrates the initial GL accounting that occurs when a collection fee is levied.

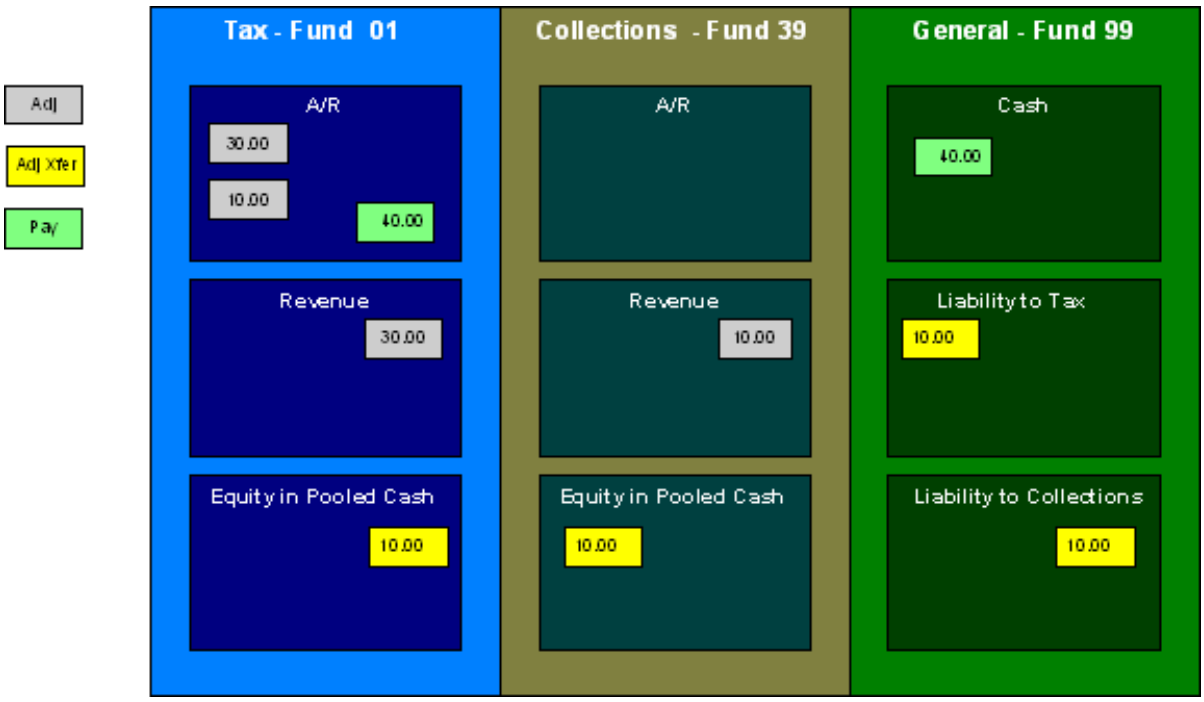


Because A/R is added to the tax fund and the revenue is added to the collections fund, the funds are out of balance.

As the tax authority takes on the responsibility to collect the fee, additional entries are created to balance the funds.



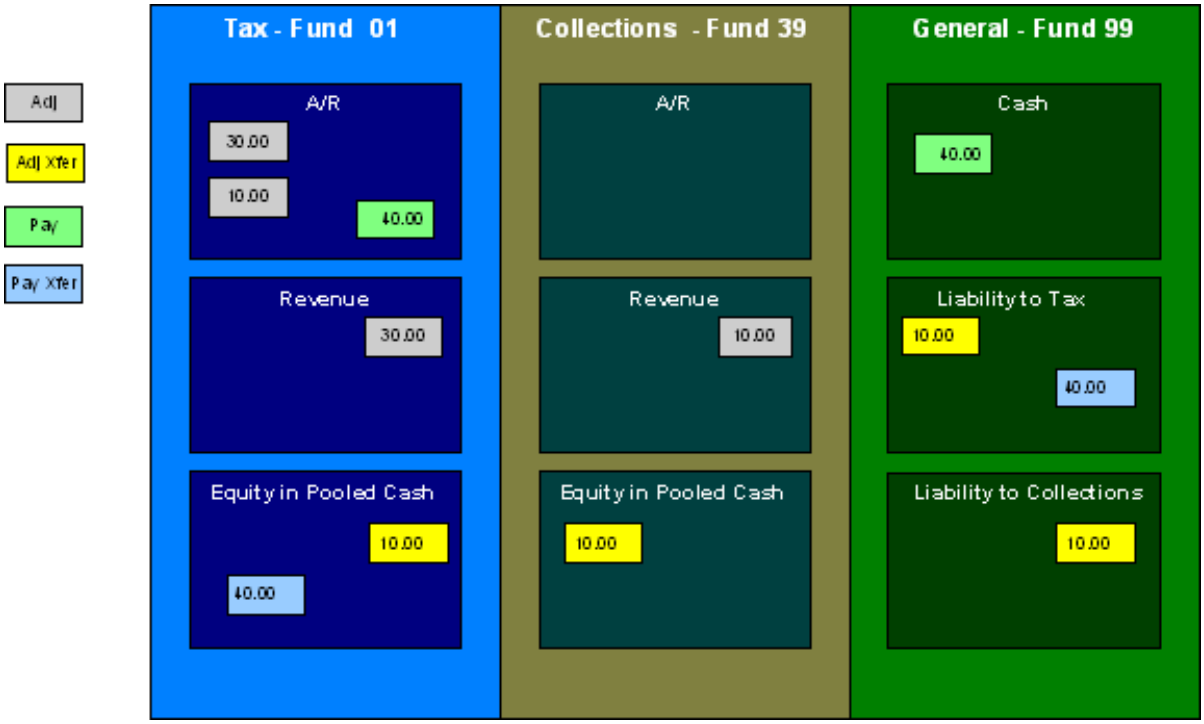
The following diagram illustrates the initial GL accounting that would occur when the payment arrives.



The tax authority's general cash account is debited, and the tax fund's A/R account is credited.

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 tax fund, causing the general fund to have an excess debit and the tax fund to have an excess credit.

From an organizational viewpoint, to make the tax fund whole, the tax fund needs to note what portion of the cash it owns, and correspondingly, the tax authority needs to note what portion of the cash is owed to each fund. The following diagram illustrates this point.



To maintain a balance of debits and credits within each fund, the tax and collection funds have an "equity in pooled cash" (EPC) account and the general fund has a liability account for each fund. In addition to debiting the general fund's cash account and crediting the tax fund's A/R accounts, the tax and collection 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

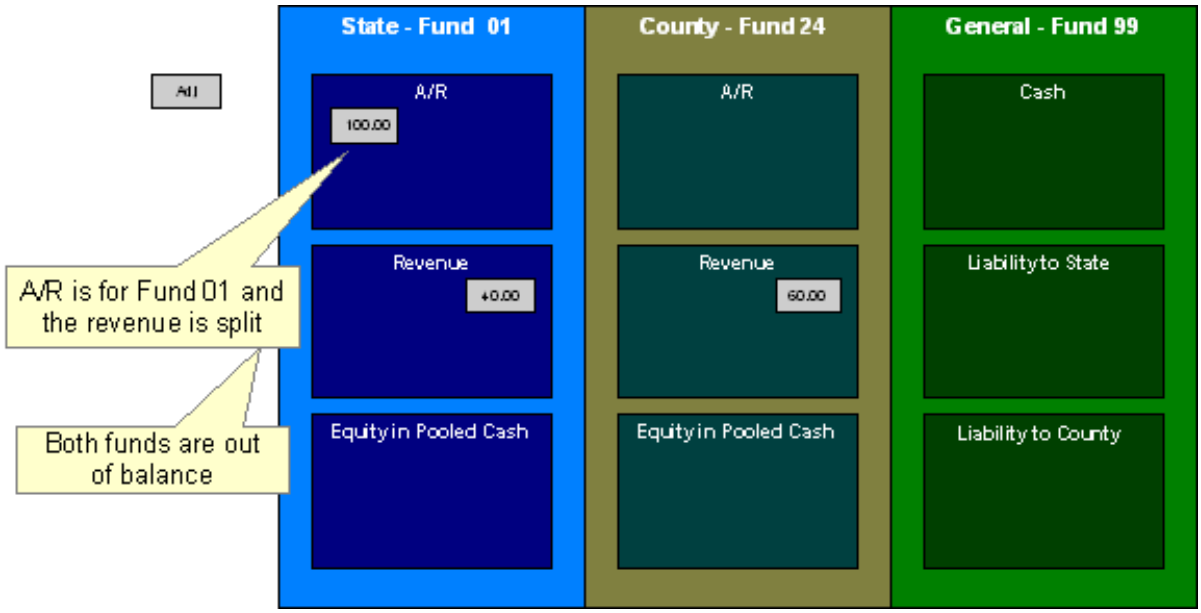
Consider a tax liability that is distributed to multiple jurisdictions where 60% is allocated to the county where the store is located and 40% is allocated to the state.

Note that in this example only two jurisdictions are used; however, a real scenario may break this down further into town, school districts, and so on.

For this example, three funds exist:

- State (fund 01)
- County (fund 24)
- General (fund 99)

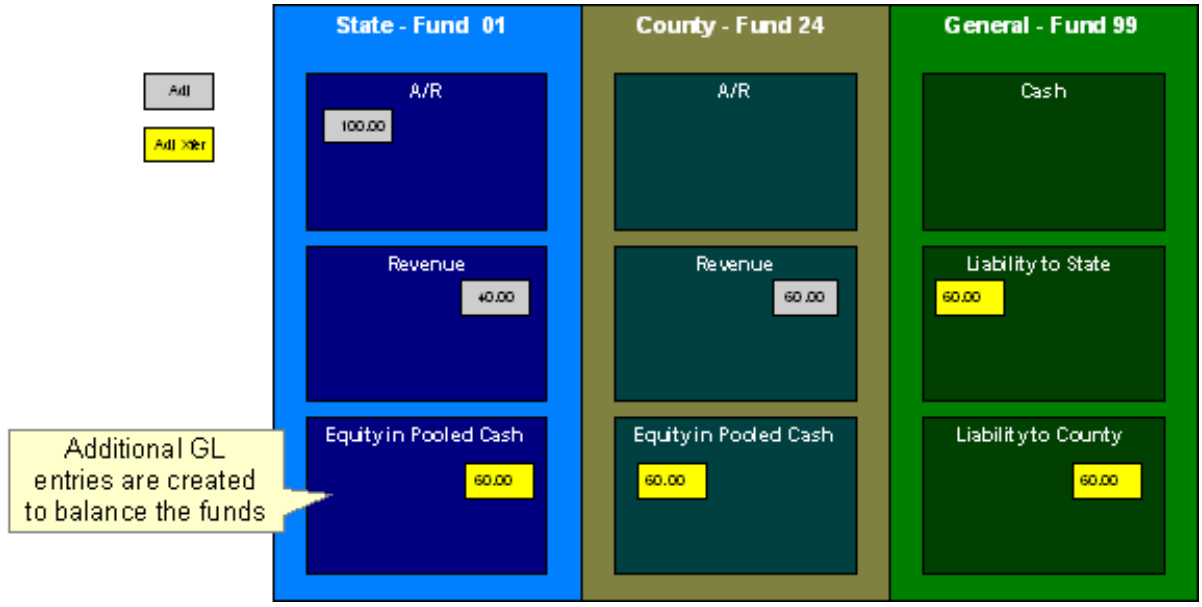
When a $100.00 tax is levied, the following diagram illustrates the initial GL entries for this scenario.



The state fund's A/R is debited, and the state and county funds' revenue accounts are credited.

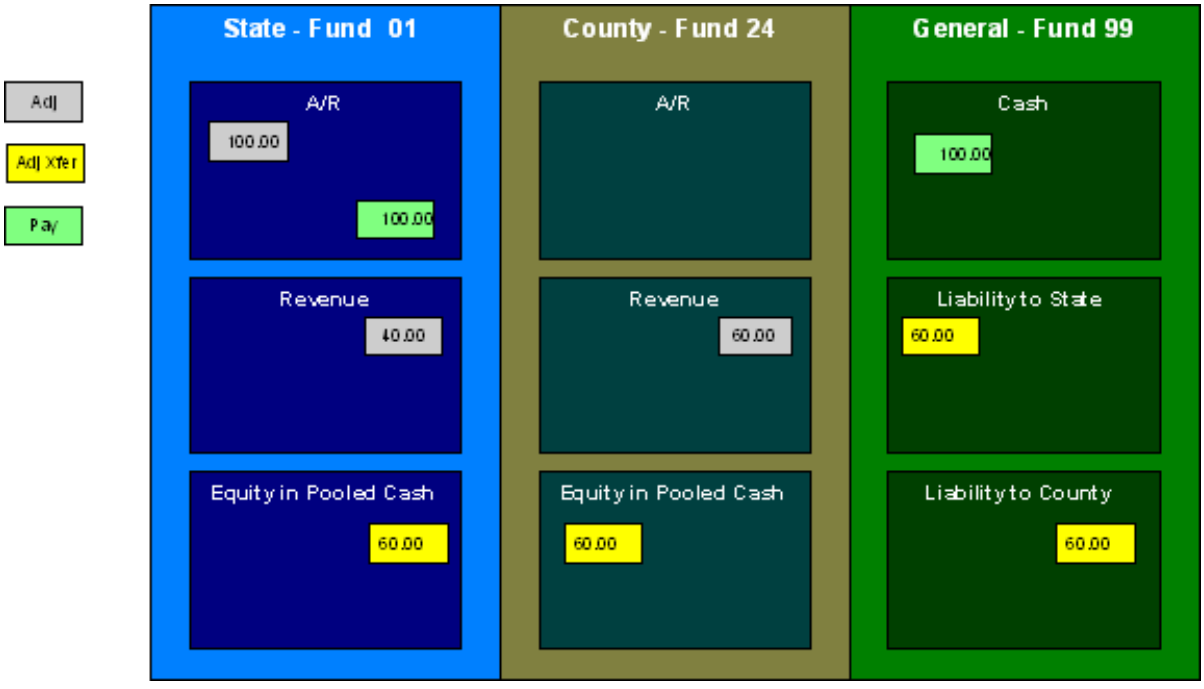
If left at this, the funds would be out of balance; the state fund would have an overall excess debit and the county fund would have an excess credit.

To balance the funds, the state fund accepts the responsibility for collecting the county taxes from the taxpayer but immediately remunerates the charges to the county fund.

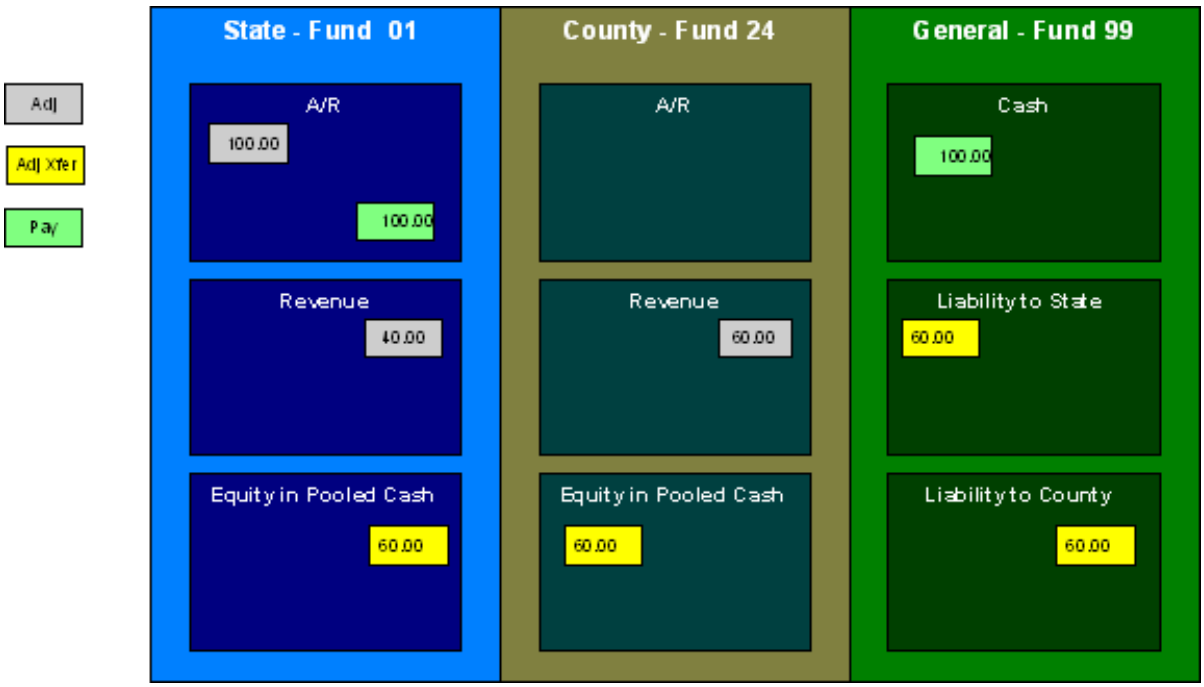


This transfer is done using the general fund. The state fund's EPC account is credited and the liability to state is debited with the amount of the county sales tax revenue. Also, the county fund's EPC account is debited and the general fund's liability to the county account is credited by the county sales tax revenue. In effect, the state fund owes the county charges to the general fund, and the general fund owes the county charges to the county fund.

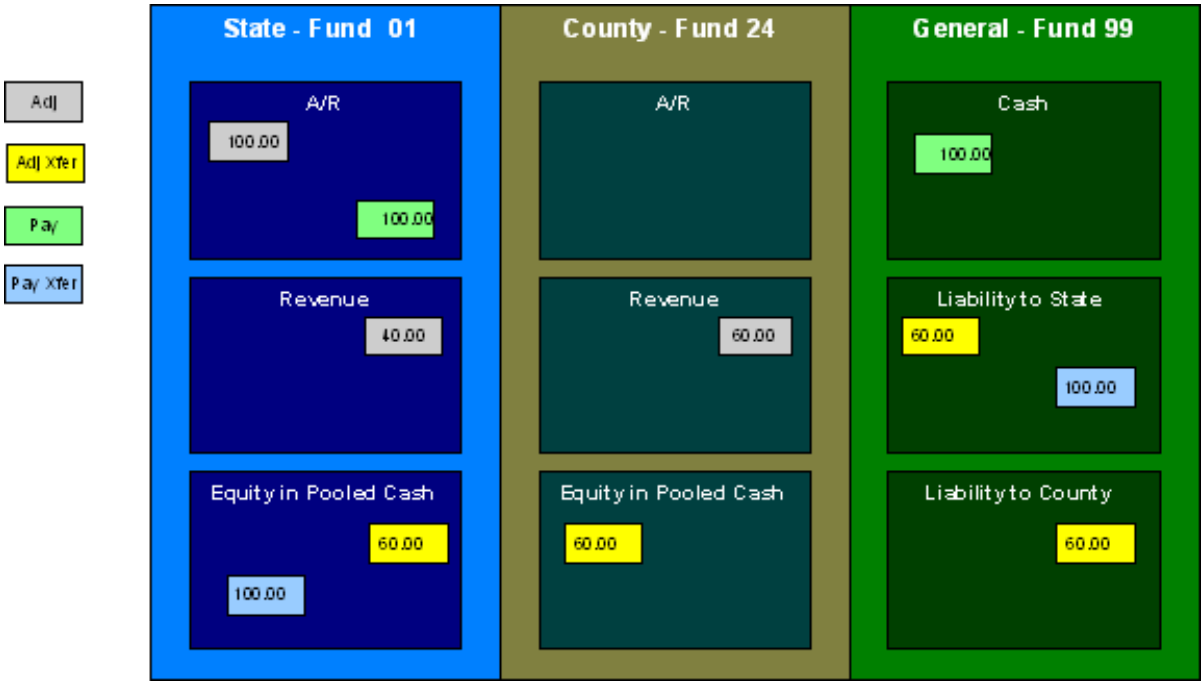
The following diagram illustrates the initial GL accounting that would occur when the payment arrives.



When the payment arrives, the cash is debited to the general fund's cash account, and the state fund's A/R is relieved. Again, the funds would be unbalanced if left in this condition; the state 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 state fund is credited by the amount of the fund's share of the cash, and the state fund's EPC is debited. Note that the payment has no effect on county fund's EPC and the general fund's liability to the county fund. The county fund "received" its money from the state department when the adjustment 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, with fund accounting activated, 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.

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**. This is the default setting.

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 liability distribution code should belong to the general fund.

Distribution Codes Must Include Fund ID

All of your distribution codes must include their respective fund ID.

➤ **Fastpath:**

For more information, refer to *Setting Up Distribution Codes*.

Update Your Funds With Their Respective Equity and Liability Distribution Codes

After distribution codes have been setup, you must update your funds to indicate the equity and liability accounts used to balance inter-fund financial transactions.

Defining Taxpayer Options

The definition of a taxpayer is someone (or something) with financial obligations with your tax authority. Within the documentation, the terms "taxpayer" and "customer" may be used interchangeably.

The system subdivides taxpayer information into the following records:

- **Person.** The person record holds demographic information about your taxpayers and every other individual or business with which your tax authority has contact. For example, in addition to normal registered taxpayers, person records may include contacts, accountants, related parties, mortgage brokers, related businesses in a corporate structure, overseas entities, persons of interest, and so on.
- **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.
- **Tax Role.** A tax role is used to define an instance of a specific tax type for a specific account. It includes information that is specific to the tax type for the account, for example, the dates that the tax is applicable for the account and the filing calendar that defines the filing frequency.
- **Obligation.** An obligation is a contract between the tax authority and the taxpayer. For example, an obligation may represent a specific filing period where a taxpayer is expected to file a return for a given tax type. An obligation may represent an ongoing charge such as a tax preparer fee. An obligation may be used to hold miscellaneous financial information such as an excess credit to later be applied to unpaid tax or debt from a third party source.

Before you can define persons, accounts, tax roles and obligations, you must set up the control tables defined in this section.

➤ **Note:**

In this section, we limit the discussion to tables that control basic demographic and financial information about your taxpayers. Other sections describe the tables that control other taxpayer related functions like forms processing, penalty and interest and collections.

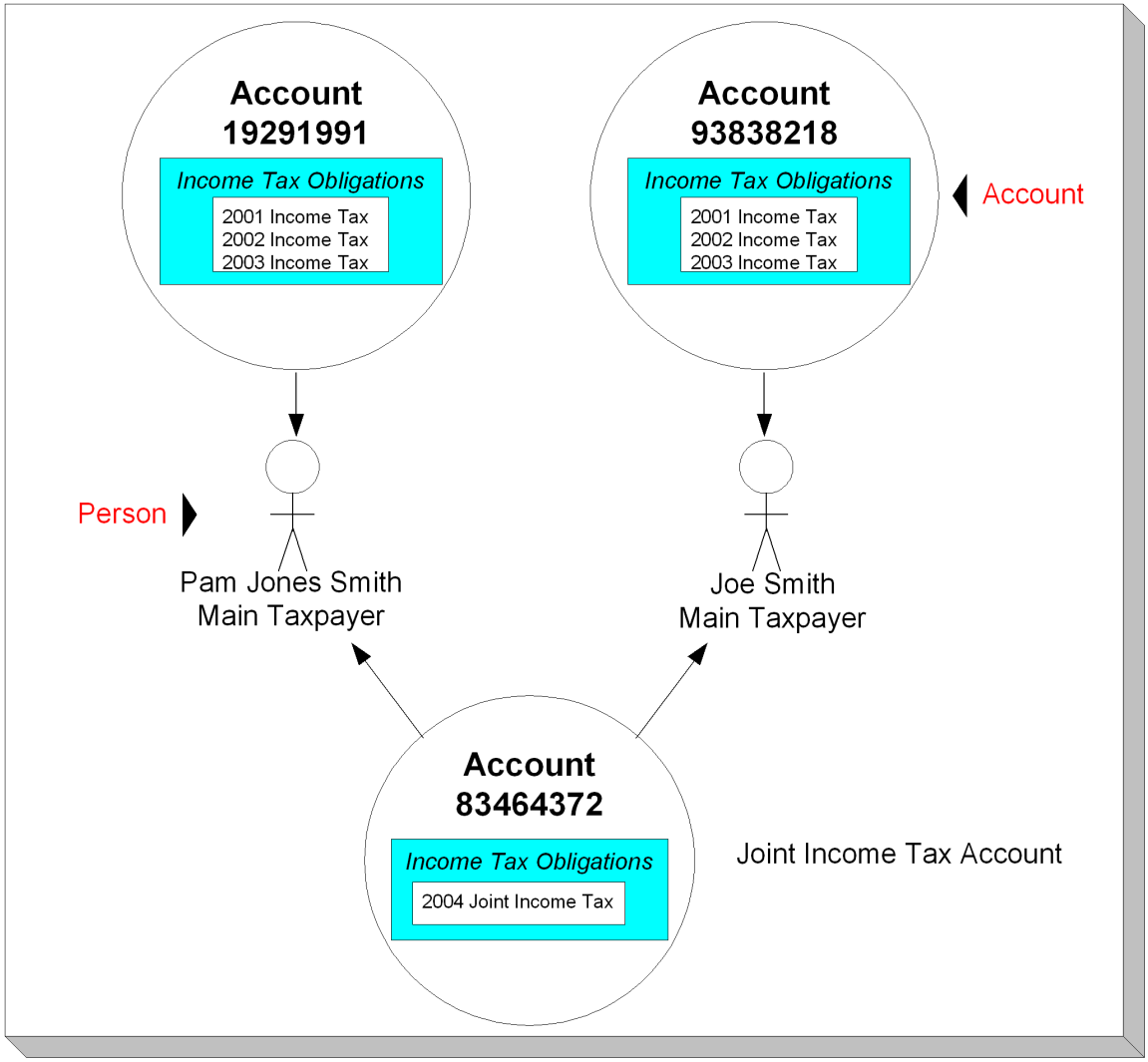
Contents

Taxpayer Overview

This section describes how the person, account, and obligation records are used to record your taxpayers' demographic and billing options.

A Simple Example Of Joint Taxpayers

The following picture illustrates two joint taxpayers: Joe Smith and Pam Jones Smith. Joe and Pam are the "main taxpayers" on their own Income Tax Accounts with their own obligations for tax years 2001, 2002, and 2003. Joe and Pam get married and file a joint tax return for tax year 2004. A new account is established for their 2004 obligation. They are both financially responsible for tax year 2004 while they each remain responsible for their prior tax years. Joe and Pam are each linked to two accounts for tax years 2001-2003 and to the new joint 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, employment information, etc.

In the above example, 2 person records would be needed, one for Pam Jones and another for Joe Smith.

A new person is added when you first have contact with a person; the person does not have to be a taxpayer before he or she is added. For example:

- If John Doe is the main contact for XYZ Corporation, John might not live in the taxing jurisdiction. You can add John as a new person and link him to any of the XYZ Corporation accounts as a contact (but not financially responsible).
- In situations with non-filers and investigations, you can add a new person and associate it with the case. If the tax authority thinks that ZZZ Corporation is operating as a business but not registered and paying taxes, the tax authority may create a person record at the beginning of the investigation to send correspondence.

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

The system allows for a taxpayer to define one account for multiple types of taxes. For example, a corporation may have a single account for the corporate income tax, sales and use tax, withholding tax, various excise taxes, etc.

Alternatively separate accounts may be used for each type of tax.

Account ID Is Non-Intelligent

The unique number of an account is referred to as the "account ID". You are probably very comfortable with this concept. You may, however, have difficulty dealing with the fact that the account id in this system has no intelligence built into it (e.g., many systems include the bill cycle and geographic location in the account id). In this system, the account ID is a random, system-assigned value.

Because the account ID contains no meaning, it can remain with a taxpayer 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 taxpayer 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 taxpayer), 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?

There a several instances when accounts must be created in the system. An account is created when:

- The taxpayer is registered for a tax obligation, or has any financial interactions with the tax authority.
- The taxpayer is registered for a new tax type. The account is created to track the obligations of the taxpayer to file or pay taxes.
- An estimated payment is received from a taxpayer for a tax that is not yet registered. The account must be created to hold the estimated payment.
- A non-filer is identified, and the tax authority decides to send a default tax assessment or penalty.

When Is An Account Expired?

Accounts never expire. Once a taxpayer has an account, the account remains in the system forever. Linked to the account are obligation records that define a taxpayer's obligation to file a tax return, and/or make a payment to the tax authority. When an account has obligations, the system pursues unpaid and overdue debt, refunds credits overdue to the taxpayer, and pursues obligations to file returns that have not been fulfilled. If the account doesn't have active obligations, the system will not produce a bill for it. You can think of an account without active obligations as being "dormant", waiting for the day when the taxpayer again files a tax return. If the taxpayer never re-files, the account (along with its financial history) remains dormant forever.

Tax Roles

Tax Roles represent the ongoing obligations of a given tax type within an account at a given point in time. It may be also be thought of as the reason a taxpayer must interact with the tax authority, or one of the taxpayer's relationships with the tax authority.

Tax roles include the dates that the tax is effective. If you consider a business, some tax types are required for the entire time a company is in business, such as corporate income tax. Other tax types may only be in effect if the company is engaging in certain activities, which may change over time.

When Is A Tax Role Created?

Most tax roles are created when the tax authority is aware that a tax type is applicable for an account. For example, when a business taxpayer registers their business and indicates the various tax types that are applicable or when an individual taxpayer files a tax form.

Filing Calendars for Filing Tax Types

For tax types that are filing period based, the onus is on the taxpayer to file the return and pay the appropriate assessment on time. The tax type typically defines the valid filing calendar(s) that govern the frequency of filing and the dates to file for each filing period. For some tax types a single calendar is valid for all tax roles of that tax type. For other tax types, one or more different filing frequencies (and therefore filing calendars) may be applicable and may change over the life of the tax role for an account.

Obligations

An obligation is a contract (either formal or implied) between the tax authority and a taxpayer. An obligation is linked to an account. There is no limit to the number of obligations that may be linked to an account.

When Is An Obligation Created?

For some tax types there is an ongoing obligation to file, such as sales tax. For obligations related to these types of taxes, it may be beneficial to include logic to create future obligations based on the start date, end date, and frequency of the obligation.

For other tax types, such as individual income taxes and excise taxes, obligation records may be created when the tax authority receives a tax return.

Due Dates for Filing Period Based Obligations

For tax types that are filing period based, the onus is on the taxpayer to file the return and pay the appropriate assessment on time. This section describes the configuration provided for defining and determining due dates.

When should the taxpayer file the return for a given filing period? When defining filing periods for a *filing calendar* you also define the statutory Due Date of the return along with the Grace Date. For taxpayers that request an extension to their filing date, an Override Filing Due Date may be defined on the specific *obligation*.

Typically the statutory payment due date for a filing period is the same date as the statutory filing due date. However, the dates may be extended independently. Extensions to a payment date are also defined on a specific obligation using the Override Payment Due Date. Also note that the grace date for the payment due date is calculated differently. Instead of using the explicit Grace Date on the filing calendar, the obligation type defines a Payment Grace Days. The logic in the base algorithms that determine if a payment is made on time (whether it be for the statutory due date or the override payment due date) considers the grace days in its calculations.

Financial Transactions Are Linked To Obligations

Fastpath:

For more information about how financial transactions are linked to obligations, refer to *The Financial Big Picture*.

Setting Up Person Options

This section describes tables that must be set up before you can define persons.

Contents

Defining Identifier Types

When you set up a person, you may define the various types of identification associated with the person, e.g., their driver's license number, their tax identity, etc. Every piece of identification associated with a person has an identification type. These identifier type codes are defined using **Admin Menu, Identifier Type**.

Note:

How are person identifiers used? The reason why identifiers are defined on a person is so that users you can look for a taxpayer using one of their person identifiers (see *Control Central - Search Facilities* for more

information). In addition, person identifiers help prevent duplicate persons from being added to the database. This is because the system warns a user before they add a new person when a person exists with the same identifier.

Note:

Person identifier types are optional. An *installation option* controls whether at least one identifier type is required on every person.

Description of Page

Enter an easily recognizable **ID Type** and **Description** for the Identifier Type.

If the identifier type has a format against which validation can be performed, use **Identifier Format** to define the algorithm. To do this:

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

Where Used

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

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*.

Defining Person Types

A person exists for every individual or business with which your tax authority has contact. Besides individual and business taxpayers, persons exist for contractors, accountants at corporate taxpayers, third party guarantors, collection agencies, etc.

The person type is used to define business rules for different types of persons. For example, for a business that is a limited liability partnership, you may wish to require that multiple individual partners be linked to the partnership.

A person type is governed by a *business object*. The base product provides business objects that you may use or extend. Refer to the BO definitions in the application. Or you may create your own.

Open **Admin Menu, Person Type** to define the person types used to categorize your persons.

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

Person Type List

The Person Type *List zone* lists every person type. The following functions are available:

- Click the *broadcast* icon to open other zones that contain more information about the adjacent person type.
- The standard actions of **Edit, Duplicate** and **Delete** are available for each person type.

Click the **Add** link in the zone's title bar to add a new person type.

Person Type Actions

This is a standard actions zone. The Edit, Delete and Duplicate actions are available.

Person Type

The Person Type zone contains display-only information about a person type. This zone appears when a person type has been broadcast from the Person Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

Defining Person Information

The person information displayed throughout the system is controlled by a plug-in.

The system first looks to see if the person type references a person business object and if the business object defines an information plug-in. If a BO is not provided or if that BO does not define an information algorithm, the system looks for an information algorithm plugged into the person maintenance object.

If no plug-ins are found on the BO or the MO, the system looks for a Person Information plug-in algorithm on the *installation record*.

If you prefer different formatting logic, your implementation should provide a plug-in at one of the above plug-in spots, as appropriate for your business rules.

Setting Up Account Options

This section describes tables that must be set up before an account can receive a bill.

Setting Up Account Management Groups

Users are informed that something requires their attention by entries that appear in To Do lists. For example, consider what happens when:

- A tax return fails validation for a math error.
- A tax return meets the business rule criteria to trigger a desk audit.
- A taxpayer refund exceeds the business rule review thresholds.

- A letter to a taxpayer is in error because the taxpayer does not have an address record.

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 **ToDo 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 open the data dictionary where you can view the tables that reference *CI_ACCT_MGMT_GR*.

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 taxpayer's bill:

Priority controls the order in which the message appears when multiple messages appear on a bill.

Note:

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 taxpayer'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:

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 Account Types

When you set up an account, you must assign it an account type. The topics in this section describe the account type control table.

Account Type - Main

To set up account types, navigate to **Admin Menu, Account Type - Main**.

Description of Page

Enter a unique **Account Type** code and **Description** for every account type.

Use **Collection Class** to define the collection class that defaults onto new accounts that belong to this account type. An account's collection class may be subsequently modified if the account has special collection problems or needs.

Use an **Account Business Object** to define a *BO* that may govern additional rules related to accounts of this type.

Turn on **Business Activity Required** if obligations linked to accounts with this account type require a Business Activity description to be entered.

Turn on **Open Item Accounting** if accounts belonging to this account type are subject to open-item account. Refer to *Open Item Accounting* for a complete explanation of the significance of this switch.

Turn on **Non Tax Agency Payment** if accounts belonging to this account type are used for payments made to reduce non-tax authority debt. For example, if your tax authority sends collection notices or offsets taxpayer refunds on behalf of other agencies, such as child support, college tuition, or parking violations, you should set up the following information to accept such payments:

Create a new account type called "Non Tax Authority Customer".

- Create an obligation type for each type of non-tax authority payment that taxpayers can make. Make sure to enter a distribution code on each obligation type that references the appropriate revenue (or payable) account. Don't forget to indicate that each obligation type is not billed.
- Create an account to which you'll book such payments. Have this account reference the new account type. We recommend creating a separate account for each obligation type that you created in the previous step.
- Create and activate an obligation for the new account(s).

When someone pays for non-tax authority 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 an account type has taxpayers. You do this on the *Account Type - 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*.

Account Type - Bill Messages

When an account type has bill messages, the system will sweep these messages onto bills created for accounts belonging to the account type. Use this page to define an account type's bill messages. Navigate to **Admin Menu, Account Type, 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 account type. 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 account type bill messages on a bill during bill completion. For more information about bill messages, refer to *The Source Of Bill Messages*.

Account Type - Controls

You must define a variety of business rules for every division in which an account type has taxpayers. Open **Admin Menu, Account Type, Controls** tab to maintain this information.

Description of Page

The **Account Type Controls** scroll contains business rules governing accounts that belong to a **Division** and **Account Type**. The following fields should be defined for each **Division**:

- Use **Days Till Bill Due** to define the number of days after the bill freeze date that the taxpayer'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).
- Use **Min Compliance Review Freq (Days)** to define the maximum number of days that can elapse between the reviews of an account's debt by the *overdue monitor*. Note, a value of zero (**0**) means that accounts in this account type will be reviewed every day.
- Use **Compliance Review Grace Days** to define the number of days that algorithms may add to a given due date when requesting that an account should be reviewed by the *overdue monitor*. For example, when completing a bill, an algorithm inserts a record into the account review dates collection setting the review date to the bill due date plus the number entered here.

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:

Warning! These algorithms are typically significant system processes. The absence of an algorithm may prevent the system from operating correctly.

You can define algorithms for the following **System Events**:

System Event	***Optional / Required***	***Description***
Autopay Amount Over Limit	Optional	This algorithm is called to handle the situation when a system-initiated *automatic payment* is created that exceeds the taxpayer'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.
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 account type / 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.
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.
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. **Only One Algorithm**. Only one algorithm to levy an NSF charge may be defined for an account type / division combination. Click *here* to see the algorithm types available for this system event.
Overpayment Distribution	Required	When a taxpayer pays more than they owe, this algorithm is called to determine what to do with the excess funds. Refer to *Overpayment Obligations* for a description on how to configure the system to handle your overpayment requirements. **Only One Algorithm**. Only one overpayment distribution algorithm may be defined for an account type / 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 account type's Days Till Due. If you need to override this method for accounts in a specific account type, specify the appropriate algorithm here.

		Only One Algorithm. Only one due date override algorithm may be defined for an account type / 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 obligations. Refer to *Payment Distribution* for more information about how payment distribution works. **Only One Algorithm**. Only one payment distribution algorithm may be defined for an account type / division combination. Click *here* to see the algorithm types available for this system event.
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 an account type has algorithms of this type, they are called after the completion of a bill for an account linked to this account type. 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 an account type has algorithms of this type, they are called immediately before completion starts for an account linked to this account type. 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 taxpayer (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 obligations. 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.

Setting Up Collection Classes

➤ **Fastpath:**

Refer to *Setting Up Collection Classes* for a description of how to set up collection classes.

Defining Account Information

The account information displayed throughout the system is controlled by a plug-in.

The system first looks to see if the account type references an account business object and if the business object defines an information plug-in. If a BO is not provided or if that BO does not define an information algorithm, the system looks for an information algorithm plugged into the account maintenance object.

If no plug-ins are found on the BO or the MO, the system looks for an Account Information plug-in algorithm on the *installation record*.

If you prefer different formatting logic, your implementation should provide a plug-in at one of the above plug-in spots, as appropriate for your business rules.

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 taxpayer (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**.

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.
- Use a **Customer Contact Business Object** to define a *BO* that may govern additional rules related to customer contacts of this type.
- 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 Filing Calendars

Filing calendars are used to define the expected filing period dates for a given type of obligation. The valid filing calendar that governs an obligation's filing period is typically defined on the obligation's *Tax Role* or its *Tax Type*.

To set up a filing calendar and its filing periods, open **Admin Menu, Filing Calendar**.

Description of Page

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

Use **Calendar Type** to categorize your calendar. For example, you may use calendar type to define the frequency like Monthly or Quarterly. You may also choose to use calendar type to distinguish special calendars like fiscal calendars. No base values are provided for this field.

Note:

Note. Define values for this field using the Lookup table. This field name is FILING_CAL_TYPE_FLG.

Specify the **Begin Date, End Date**, **Due Date** and **Grace Date** for each filing period. Refer to *Due Dates for Filing Period Based Obligations* for more information about due dates.

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_FILING_CAL*.

Setting Up Tax Types

The tax type is used to defined business rules for each type of tax you support. Tax type is required when defining any financial obligation so a catchall tax type of **Other** for write-offs, payment plans and other miscellaneous uses may be warranted.

A tax type is governed by a *business object*. The base product provides business objects that you may use or extend. Refer to the BO definitions in the application. Or you may create your own.

Open **Admin Menu, Tax Type** to define the tax types used to define information about the tax types you support.

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

Tax Type List

The Tax Type *List zone* lists every tax type. The following functions are available:

- Click the *broadcast* icon to open other zones that contain more information about the adjacent tax type.
- The standard actions of **Edit, Duplicate** and **Delete** are available for each tax type.

Click the **Add** link in the zone's title bar to add a new tax type.

Tax Type Actions

This is a standard actions zone. The Edit, Delete and Duplicate actions are available.

Tax Type

The Tax Type zone contains display-only information about a tax type. This zone appears when a tax type has been broadcast from the Tax Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

Setting Up Obligation Options

This section describes tables that must be set up before you can define obligations.

Setting Up Industry Codes

An obligation can reference an industry code. This code is used to categorize obligations for reporting purposes. To define an industry code, open **Admin Menu, Industry Code**.

Description of Page

Enter a unique **Industry Code** and **Description**.

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 obligations 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 (obligations) have contractual limits. For example, a vehicle tax account may have special considerations or limits on the amount it can be billed, such as vehicles for government agencies or diplomats. The obligations 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*.

Obligation Type Controls Everything

Every obligation references an obligation type. The obligation type controls all aspects of an obligation's behavior including 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 obligation 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 obligation types.

Fastpath:

For more information about obligation types, refer to *Defining Obligation Types*.

Financial Controls

There are also a number of control tables that must be set up to control the bills, payments, and adjustments that are linked to an obligation. For more information about these tables, please refer to *Defining Financial Transaction Options*.

Defining Property Options

This chapter describes options related to physical locations and to property owned by a taxpayer. A location is the physical location associated with an asset, such as a vehicle or building, that a taxpayer pays taxes on. The following sections describe the control tables needed to support this functionality.

Setting Up Location Options

This section describes tables that must be set up before you can define locations.

Defining Location Types

Open **Admin Menu** > **Location Type** to define the location types used to categorize your locations.

Description of Page

Enter a unique **Location Type** and a **Description** for every location type.

Use a **Location Business Object** to define a *BO* that may govern additional rules related to locations of this type.

Where Used

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

Defining Location Information

The location information displayed throughout the system is controlled by a plug-in.

The system first looks to see if the location type references a location business object and if the business object defines an information plug-in. If a BO is not provided or if that BO does not define an information algorithm, the system looks for an information algorithm plugged into the location maintenance object.

If no plug-ins are found on the BO or the MO, the system looks for a Location Information plug-in algorithm on the *installation record*.

If you prefer different formatting logic, your implementation should provide a plug-in at one of the above plug-in spots, as appropriate for your business rules.

Setting Up Location Postal Defaults

You set up postal defaults if your revenue authority is able to default field values onto new locations based on the location's postal code. The topics in this section describe how to maintain postal defaults.

➤ **Fastpath:**

For more information about where these default values are used, refer to *Maintaining Locations*.

Postal Defaults - Main

To define location postal defaults, open **Admin Menu** > **Postal Code Default** .

Description of Page

Enter the **Country Code** and range of postal codes to which the default values apply using the **From Postal Code** and **To Postal Code**.

➤ **Note:**

Note. You may not have postal defaults whose from / to postal codes overlap.

Enter the **County** to be defaulted onto new locations located in this postal code range.

Enter the **City** to be defaulted onto new locations located in this postal code range.

Enter the **Division** to be defaulted onto new locations located in this postal code range.

Enter the **State** to be defaulted onto new locations located in this postal code range.

Enter the **Time Zone** to be defaulted onto new locations located in this postal code range.

Use the **Characteristic Types and Values** collection to define the **Characteristic Type**s and their respective **Characteristic Value**s to be defaulted on locations located in this postal code range. In addition to providing interesting information, these characteristics may also determine the prices and rates on the bills generated for properties associated with this location.

Fastpath:

For more information about characteristics, see *Setting Up Characteristic Types & Their Values* and *An Illustration Of A Rate Factor And Its Characteristics*.

Use the **Geographic Types and Values** collection to define the **Geographic Type**s and their respective **Value**s to be defaulted on locations located in this postal code range. In addition to providing interesting information, these values may be used to sort field activities in geographic value order.

Where Used

This information is defaulted when a new location is added. Characteristics and geographic values are also defaulted when the postal code for a location is changed. Refer to *Maintaining Locations* for more information.

Defining Forms Processing Options

Forms processing is a core function for a tax authority.

Registration forms are used for taxpayer registration and / or taxpayer demographic information updates. These forms are processed in the context of a taxpayer. When a registration form is processed, it usually results in the creation of new taxpayers / accounts / tax roles / obligations or updates to existing taxpayer information - e.g. name / address change.

Tax forms are the most common types of forms. These forms are processed in the context of a filing period. When a tax form is filed, it satisfies an obligation to file and its processing results in an assessment.

Most forms have a common structure. The form consists of one or more sections, with each section containing one or more lines or groups of lines. Processing rules can be associated with any of these components.

As form changes are introduced from year to year, the changes usually entail adding or removing specific sections and / or lines, as well as adding / removing the processing rules associated with them.

Forms can come from one or more sources. They can be interfaced in batch from an external system or created internally.

The Big Picture of Registration

Registration forms are used for maintaining taxpayer demographic information. Examples of these forms are a business registration form and a change of address form.

A taxpayer is registered in any of the following ways:

- The taxpayer files a paper registration form
- The taxpayer submits a registration form online.
- A tax authority user registers the taxpayer using an online registration form
- For some tax types, the system can automatically register the taxpayer when the first tax form is processed.

Registration Forms Are Linked to Taxpayers

Registration forms create / update taxpayer information. When a registration form posts, it may include actions like adding / updating accounts, tax roles and obligations.

Unlike tax forms, registration forms typically don't result in assessments. Thus, these forms usually do not have a financial effect.

Registration Forms Have A Common Lifecycle

Most registration forms go through a similar lifecycle:

The registration form is created in an initial state where processing rules are not yet executed. This allows the form to be saved as work-in-progress.

Validated	A validation step verifies the information on the registration form. Any issues detected at this step usually causes form processing to stop.
Suspense	A registration form goes into this state when the validation step detects errors on the form that a user may be able to investigate and resolve. Exceptions are stored in the system so that a user can resolve the issues accordingly. The form needs to be re-validated after the issues are resolved.
Waiting for Information	A registration form goes into this state when the validation step detects missing information, thus preventing the system from further proceeding with form processing. Exceptions are stored in the system so that a user can track the issues accordingly. In addition, this state usually triggers correspondence to the taxpayer, requesting for the missing information. The form is usually re-validated after the missing information is received.
Posting	When a tax form passes validation, it is posted in the system. Actions can include creating the taxpayer / account / tax role / obligation, confirming registrations, etc.

The Big Picture of Tax Forms

A tax form fulfills an obligation to file a tax return for a specific filing period. When a tax form is processed, it results in an assessment, which determines either a tax amount due or an overpayment.

Tax Forms Are Linked to Obligations

When a tax form is posted, it is associated with an obligation for a specific filing period. Any financial transactions relating to the filing period will be created under this obligation. The obligation can be created in advance in anticipation of filing, such as in the case of business taxes or created at the time the form is posted, such as in the case of individual income tax.

The form type references one or more obligation types, and this information is used when determining what type of obligation to create for a specific tax form. Typically a form type would only reference one obligation type. For

example, taxes such as sales and use, individual income, and corporate income would typically have one obligation type defined per form type. The base business object **C1-StandardTaxFormType** supports the definition of one obligation type on a form type.

There are situations where a form type would be associated with more than one obligation type. These include business activity statements, estimated payments, audit forms. A different business object can be used to support these situations. Alternatively, the validation on the base business object can be disabled to allow more than one obligation type for a form type.

The obligation types defined for a form type are used to restrict the form type list for a specific tax type, when a user adds a tax form.

Tax Forms Have A Common Lifecycle

Most tax forms go through a similar lifecycle.

Pending	The tax form is created in an initial state where processing rules are not yet executed. This allows the form to be saved as work-in-progress.
Validation	A validation step verifies the information on the tax form. Any issues detected at this step usually causes form processing to stop.
Waiting for Information	A tax form goes into this state when the validation step detects missing information, thus preventing the system from further proceeding with form processing. Exceptions are stored in the system so that a user can track the issues accordingly. In addition, this state usually triggers correspondence to the taxpayer, requesting for the missing information. The form is usually re-validated after the missing information is received.
Posting	When a tax form passes validation, it is posted in the system. Actions can include creating the taxpayer / account / tax role / obligation, creating financial transactions, etc.
Suspense	A tax form goes into this state when the validation step detects errors on the form that a user may be able to investigate and resolve. Exceptions are stored in the system so that a user can resolve the issues accordingly. The form needs to be re-validated after the issues are resolved.
Adjust	A posted tax form can be adjusted for any of the following common reasons: • Correcting data capture errors or data entry errors • Line item adjustments initiated by the taxpayer • Adjustments initiated from an audit When a tax form is adjusted, the financial effect of the existing tax form is canceled and new financial transactions are created for the updated line items.

Transfer	A tax form can be transferred to a different taxpayer/obligation if it previously posted to the incorrect taxpayer/obligation. When a tax form is transferred, the financial effect of the existing tax form is removed from the current obligation and new financial transactions are created for the 'transfer to' obligation.
Reverse	Some exceptional cases may require that the financial effect of a tax form be canceled, without necessarily creating new financial transactions. A tax form reversal is a way of voiding a tax form that has already posted.

The base package provides a sample tax form **C1-ParentTaxForm** that has a common lifecycle.

Note:

Refer to the Business Object metadata for more information.

Categorizing States

The business objects provided in the base product include a status option for the states in the lifecycle called Status Category. The values provided in base identify the states in which the form is considered **In Effect** and the states in which it is considered **Not In Effect**. This setting may be used in code to allow or prevent certain actions from occurring based on the state of the form. For example, a user may not cancel an Obligation if there are any tax forms for the obligation whose state is **In Effect**.

In addition, the **POSTED** status defines an additional status category value for **Posted**, allowing for validation or processing based on whether a form is posted or not.

This option allows multiple values to exist for a given state.

Other Changes After a Tax Form Posts

The following sections describe other possible changes after a tax form posts.

Standalone Amendment Forms	A posted tax form can also be adjusted by processing a standalone amendment form. In this case, the amendment form is processed independently of the original form. New financial transactions are created for the difference between the financial effect of the original form and the financial effect of the amendment form.
Standalone Audit Forms	A standalone audit form can also adjust a previously posted tax form. In this case, the audit form is processed independently of the original form. New financial transactions are created for the difference between the financial effect of the original form and the financial effect of the audit form.

Payments Related to Tax Forms

Some tax types require taxpayers to make estimated payments based on specific conditions - e.g. estimated revenue, estimated tax liability, etc. The system allows for payments to be made in advance. These payments are usually reconciled when the associated tax forms are processed.

The system also caters for payments sent together with tax forms. The payments are usually created when the form posts, suspends or cancels (depending on the situation).

Payments in Advance / Estimated Payments

Some tax types require taxpayers to make estimated payments based on specific conditions -e.g. estimated revenue, estimated tax liability, etc.

When the payment is processed, the obligation may or may not exist yet.

- If the obligation exists, the payment is applied to that obligation
- If an obligation does not exist but the taxpayer has an existing account, the payment is applied to an excess credit obligation under that account.
- If the taxpayer does not have any accounts, the payment is applied to a company suspense account.

In the cases where the payment got applied to an excess credit obligation or company suspense, the payment needs to be transferred when the tax form is processed and the obligation is determined. To do this, a form rule needs to be configured to do this transfer at the point where the form's obligation is identified.

Fastpath:

Refer to *Defining Form Rules* for details on how to configure rules.

It's important for the payment to capture details that will facilitate the transfer of the payment to the correct obligation when the tax form gets processed. The payment should capture details like the taxpayer's identifier and the tax type / filing period being paid.

Payments with a Tax Form

When payments are sent with a tax form, the payment is usually linked to the form by a unique identifier, such as a document locator.

The payment is processed in two possible ways, which are described in the following sections

Payment Is Processed Separately From The Tax Form

This is applicable for implementations that have separate processes for payments and forms.

In this case, the payment is applied depending on which of the payment and the tax from gets processed first:

- If the tax form is processed before the payment, the payment is applied to the obligation on the form.
- If the payment is processed before the tax form and:
- The taxpayer has an existing account, the payment is applied to an excess credit obligation under that account
- The taxpayer does not have any accounts; the payment is applied to a company suspense account.

In the cases where the payment got applied to an excess credit obligation or company suspense, the payment needs to be transferred when the tax form is processed and the obligation is determined. To do this, a form rule needs to be configured to do this transfer at the point where the form's obligation is identified.

Fastpath:

Refer to *Defining Form Rules* for details on how to configure rules.

It's important for the payment to capture details that will facilitate the transfer of the payment to the correct obligation when the tax form gets processed. The payment should capture details like the taxpayer's identifier and the tax type / filing period being paid.

Payment Is Processed With The Tax Form

This usually occurs when payments are uploaded with tax forms.

In this case, the payment is not created until the form is processed. In order to process payments within tax form processing, a form rule needs to be configured.

Fastpath:

Refer to *Defining Form Rules* for details on how to configure rules.

Directing a Payment to a Form or Filing Period

The base package provides a **Pay A Form / Filing Period** Distribution Rule - Create Payment algorithm type *C1-PAYFRM*. This algorithm type assumes that the payment event distribution details can be specified as any of the following

- Document locator only. The algorithm will post the payment to the tax form's obligation. If the tax form is not found, the payment is applied to either an excess credit obligation or company suspense account.
- Document locator with taxpayer identifier number, tax type and filing period. The algorithm will post the payment to the tax form's obligation. If the tax form is not found, the payment is applied to either an excess credit obligation or company suspense account.
- Taxpayer identifier number, tax type and filing period (i.e. advanced / estimated payments). The algorithm will post the payment to either an excess credit obligation or to company suspense account.

To enable this algorithm type:

- Configure an algorithm for C1-PAYFRM with the appropriate parameters. Use base characteristic types for DLN, Taxpayer Identifier, Tax Type and Filing Period.
- Create distribution rules for each of the distribution details. Use base characteristic types for DLN, Taxpayer Identifier, Tax Type and Filing Period.
- Plug in your Pay A Form / Filing Period algorithm in one of the distribution rules you created.

Note:

The distribution detail with the 'create payment' algorithm is last. When creating the payment event using the above distribution rules, the detail that has the 'create payment' algorithm must be specified as the last. This ensures that all details are available by the time the algorithm executes.

The Big Picture of Form Definition

Form Type Controls Form Processing

Each tax form or registration form must reference a form type. The form type may reference business objects that define:

- The structure, lifecycle, processing rules and processing options of the form type itself.
- The structure, lifecycle, processing rules and processing options of the actual tax form or registration form.

Designing Form Types

A Form Type needs to be configured for each type of form that the system needs to process.

The following are the steps in configuring a form type:

- Define basic information about the form type, including the business object that controls the behavior of the form type. The base product provides the business object **C1-StandardTaxFormType**, which is used for automatic generation of tax form business objects. Also determine the applicability of change reasons to the form type.
- Define the sections of the form.
- Define the lines that each section contains.
- Generate the form type.
- Define the valid form change reasons to the form type, if change reasons are applicable.
- Link the form type to form rule groups.

Designing Sections

When designing sections of your form, the first thing to determine is the number of sections. Small or simple forms may contain only one or two sections, while complex forms, such as forms with schedules, usually contain many sections.

Identify any repeating sections. This is common with tax forms that allow multiple occurrences of a schedule. For such recurring sections, determine whether or not recurrence should be limited to a finite number.

Designing Lines

After establishing the sections, you need to define the information that is contained in each of the sections. Some sections could include one or more simple lines. Some sections may include one or more repeating groups of lines. Other sections may have a mix of simple lines and repeating groups.

Define each line. Most lines on a tax form are either character or numeric fields. These lines could simply contain a value or could capture additional information.

The base package supports three common types of lines: Standard Line, Group Line and Label Line

Standard Lines

Most form lines fall into this category. This type of line usually contains the current value of the line.

This line type may also contain additional information that relates to the line's current values. For example, the value that was originally reported by the taxpayer (also referred to as the 'as reported' value), a reason explaining a difference between the 'as reported' and 'current values', an indication that rules processing should bypass any calculations and take the current value as-is, etc.

Refer to the definition of business object **C1-StandardLine** for more details.

Group Lines

This type of line is used for defining repeating lines in the section. This line contains header information about the group. Every line that belongs to the group refers to their header line.

Refer to the definition of business object **C1-GroupLine** for more details.

Label Lines

This line type is used for displaying text on the form.

Refer to the definition of business object **C1-LabelLine** for more details.

Assigning Business Names to Sections and Lines

Sections and lines are assigned business names to give form rules configuration a way of referencing the sections and lines, without necessarily knowing the specific form types that contain them. This allows form rules to be configured for reuse across several form types.

The actual instances of the sections and lines within a form type are determined when the rules are executed.

Fastpath:

Refer to *Defining Form Rules* for details on how to configure rules.

Designing Form Change Reasons

As part of designing your form types, you will need to decide if the form should require the user to supply a change reason when modifying a form. The form type includes the following configuration:

- **Overall Change Reason Applicability** - this controls whether change reason is used at the overall form level.
- Set this to **Required** if the user needs to enter a change reason when making changes to a form.
- Set this to **Optional** if the user can choose to enter a change reason but is not required.
- Set this to **Not Applicable** if this form does not use overall change reasons. This will suppress the change reasons grid and comments from the main section of the form's UI map.
- **Line Change Reason Applicability** - this controls whether change reasons are used at the individual form line level.
- Set this to **Applicable** if this form uses change reasons at the form line level for one or more of its form lines. The change reasons definition checkboxes will be enabled when defining a form line.
- Set this to **Not Applicable** if this form does not allow change reasons at the form line level. The change reason definition checkboxes will be disabled when defining a form line.

  In addition to controlling which change reasons fields appear on the form's UI map, the applicability flags also control the validation for change reasons on the form. See *Determining When a Change Reason is Required* for more information.

  If form change reasons are going to be used with this form type, then a list of one or more change reasons needs to be defined. This list defines the valid change reasons for the form type and is used to create the dropdowns on the form for the user to choose from.

  Some change reasons are used by algorithms that update a form, rather than a user. These change reasons will have a system default flag associated with them and will not appear on the dropdown presented to the user. A lookup defines which system transitions can be used. Additional values can be added by an implementation as needed.

  The base package provides the algorithm type *C1-DFLTFCR* that defaults a change reason when a form is suspended or is waiting for information. If your form's business object is using the business object **C1-ParentTaxForm**, you will need to have one change reason with the system default flag of **Form Transition**.

Generating Forms

The base package provides a form type business object **C1-StandardTaxFormType** that can be used for generating most tax form types. This business object's lifecycle includes a generation step and that allows for regenerations when needed.

The Lifecycle of a Form Type

A form type is created and stays in a 'pending' state while the details about the form type, its sections and its lines are being configured.

Once the structure of the form type, sections and lines are defined, the next step is to generate the form business object and its UI maps.

If form generation encounters any problems, the form type goes into an error state. The user needs to fix the issue(s) and regenerate the form.

On the other hand, if form generation is successful, the form type goes into a 'generated' state until the user activates it.

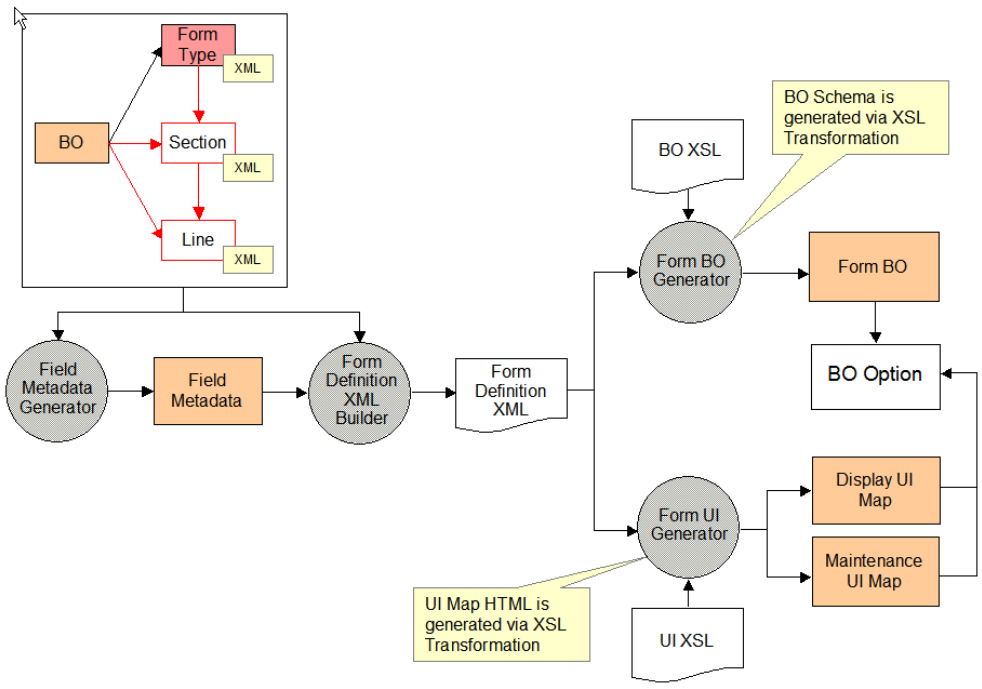
Activated forms can be inactivated.

If changes need to be made after a form type is already generated, a user could put the form type back into the 'pending' state and change / regenerate the form type accordingly.

The same could be true for activated or inactivated form types - if changes are allowed when the form type is already in either of these states.

The Form Generation Process

The following diagram illustrates the form generation process flow.



Form Generation

The steps are discussed in the following sections.

Generating the Form Type

When using the base business object **C1-StandardTaxFormType** for generating tax forms, the form generator is invoked when the form type goes into the Generated state.

An 'enter' algorithm (*C1-GEN-FORM*) invokes the following processes (in order):

- Form Business Object Generation
- Form Display Map Generation
- Form Maintenance Map Generation

All three processes have to run successfully.

If any of the processes fail, the algorithm creates an appropriate log entry to indicate which one failed. Any components that were generated prior to hitting the error are rolled back.

Form Business Object Generation

This process creates a form business object using the form definition data on the form type. It includes intermediate steps for: creating Field metadata, creating Lookup Fields / Values and extracting form definition data.

Field Metadata Creation

This process goes through the form type's section and line definitions and creates label fields, definition fields and override label fields based on the section and line configurations. For definition fields that are configured as lookups, this process has an additional step of creating the Lookup Fields and their Lookup Values.

Form Definition Extract

This process extracts the form type / section / line definitions into one XML document that feeds into both the business object creation and UI map generation processes.

Form Business Object Creation

This process creates the form business object using the form definition XML data that was extracted prior. The form business object is created as a 'child' BO that inherits from a parent form business object - as defined on the form type business object's 'Form Parent Business Object' option.

Note:

The **C1-StandardTaxFormType** references **C1-ParentTaxForm** as its 'Form Parent Business Object'.

Any common form elements should be placed in the parent form BO's schema. Other form elements that vary between form types should have corresponding Line definitions. The sections and line definition are generated into the form BO schema accordingly.

This process also creates the Application Service for the generated business object. The associated access modes are the same as the parent business object's access modes.

Note:

These generated Application Services must be configured on your user groups accordingly.

Form Display / Maintenance UI Map Generation

These processes create the form's display and maintenance UI maps and automatically link the maps to the generated form BO.

The 'Form Display/Maintenance UI XLS Stylesheet' specified on the form type business object controls how the HTMLs in these maps are generated.

Reserved Primary Keys

The form generator populates the primary keys of the generated components as follows:

- Form BO code (max length 30) = Form Type code (max length 12)
- Display UI Map code (max length 30) = Form Type code (max length 12) + 'Display'
- Maintenance UI Map code (max length 30) = Form Type code (max length 12) + 'Maint'
- BO Application Service ID (max length 20) = Form Type code (max length 12 - note: this is also the generated BO code) + 'BOAS'

If initial generation of BOs / UI maps / application services finds a conflict with any existing CM primary keys for the same objects, form generation will result in an error. Either the existing CM objects or the new Form Type code should be changed accordingly.

Regenerating a Form Type

Refer to *The Lifecycle of a Form Type* for information on when / how a form type is regenerated.

Once the form business object / UI maps / application services exist, any subsequent form type regeneration will retain the primary key, as well as most of the references associated with the generated objects - e.g. BO options, BO plug-ins, etc. Only the following updates would take place:

- The existing BO's schema is replaced.
- The existing UI maps' HTML is replaced.
- If the parent form BO referenced on the form type has changed since the last generation:
- The Parent BO and Maintenance Object references on the existing BO are replaced.
- The access modes on the child application services are replaced.

Overriding the Generated UI Maps

The look-and-feel of the generated UI maps is controlled by an XSL stylesheet that is specified on the form type business object's 'Form Display/Maintenance UI XSL Stylesheet' option.

Overriding All Generated UI Maps

If your implementation needs to override the look-and-feel of all generated UI maps, you can specify your own XSL Stylesheet on the form type business object.

If you are using the base form type business object, you need toadd a 'Form Display/Maintenance UI XSL Stylesheet' option with a higher sequence.

Overriding Specific Generated UI Maps

If your implementation needs to override the look-and-feel of specific UI maps or if you need to add implementation-specific logic to the generated UI maps, you can do the following:

- Make a CM copy of the generated UI maps and put the additional logic on those copies.
- Plug the CM UI maps into the corresponding BO option types on the generated form BO. Use a higher sequence number than the base-generated BO options. The 'UI map' BO options types are single-instance options, for which the FW knows to use the options with the highest sequence number.

Note:

Generated UI maps will use sequence number 10. CM maps must use sequence numbers higher than 10. If your CM maps do not follow this rule, you will lose your CM maps' links to the BO after regeneration.

When the BO and generated UI maps are regenerated, the form generator process simply replaces the XML in the BO schema and the HTML in the UI maps. The BO options added from the initial generation are retained. Thus, any CM UI maps would continue to be in effect.

After regeneration, your implementation must compare the generated maps with the override maps and copy the differences to the override maps accordingly.

Designing The Form's Processing Rules

The final step in configuring a form is the definition of processing rules.

Fastpath:

Refer to *Defining Form Rules* for details on how to configure rules.

The Big Picture of Form Versions

A form is likely be modified several times during its lifetime. Whenever one or more changes are made to a form, a new form version is created. The rules that determine which changes constitute a new form version vary from one tax authority to another.

In general, there are two main categories of form changes:

- User correction. These are changes made to a form by a user for a variety of reasons such as correcting a scanning error, correcting a taxpayer error, update of a mailing address resulting from a taxpayer phone call, or correction of a form's receive date.
- Auto correction. These are changes made automatically by the system for reasons such as correction of a calculation error.

The tax authority requires the ability to track changes to a form for a couple of reasons:

- Audit Trail. The need to track who made the changes to the form and when the changes were made.
- Taxpayer Inquiry. The tax authority needs to be able to explain to the taxpayer any differences between the information that the taxpayer reported and the current information on the form.

How are Form Versions Created

A form version is created when the form is modified either by a user or by a backend process (e.g. algorithm, service). The base package business object **C1-ParentTaxForm**includes two algorithms, which take a snapshot of the form that is being changed and store it in a special type of log record.

This log record, with a log type of **Form Version**, includes the details of the form version such as the user who made the change, the date and time of the change and a snapshot of the whole changed form.

The form versions will get created in the following scenarios:

- A pending form is transitioned to the next state. When the form exits the **Pending** status, the algorithm type *C1-PTF-UARI* will create the initial form version. This version is also known as the 'As Reported' version. Its purpose is to capture the form's data as reported by the taxpayer.
- A form is modified and is in a status that indicates a version should be created. The base business object includes the audit algorithm type *C1-TXCREFRMV*. This algorithm type will create a form version if the form is in a status that has the status option **Enable Versions** set to **Y** and any of the form's fields have changed.

Form Version Display

A form version is stored as a snapshot on the tax form log record. A user can view this form version on the **Tax Form - Versions** portal. This portal contains a list of the form versions for the tax form in context. A user can select a form version from the list and have it displayed below.

The form version is displayed using a derived map zone, which utilizes the same UI map used by the form itself. A derivation script looks at the form's business object options to determine which display UI map and display map service script should be used. This allows the form version to have the same look and feel as the form itself.

Form Changes Typically Require a Reason

Typically when a user makes a change to a form, they will be required to enter a reason for making the change. Some tax authorities require a user to enter a specific reason for each line that is changed, while others only require a more generic reason that describes the nature of the change for the overall form. The types of form changes supported by the base package are:

- Overall Change Reason. This change reason is specified at the form level and is used to describe the nature of the changes made to the form. Some examples include: corrected scanning error, data entry, and form rules auto correction.
- Line Change Reason. This change reason is specified at the form line level and is used to describe the specific change made to a form line. Some examples include: additional information provided by taxpayer, supporting documentation not accurate. It is possible for some form lines to be associated with a form change reason while others do not. For example, a change to an address might not require a change reason, while a change to the deductions amount will. The forms definition facility allows you to define the lines that need a change reason.

The form type controls which type(s) of change reason are applicable to the form, if any. A form type can be set up to require overall change reasons, form line change reasons, both or none. See *Designing Form Change Reasons* for more information on how a form type can be set up to use change reasons.

Determining When a Change Reason is Required

The base business object **C1-ParentTaxForm** includes a validation algorithm type *C1-PTF-VCRS*. This algorithm type uses the following conditions to determine if a change reason is required:

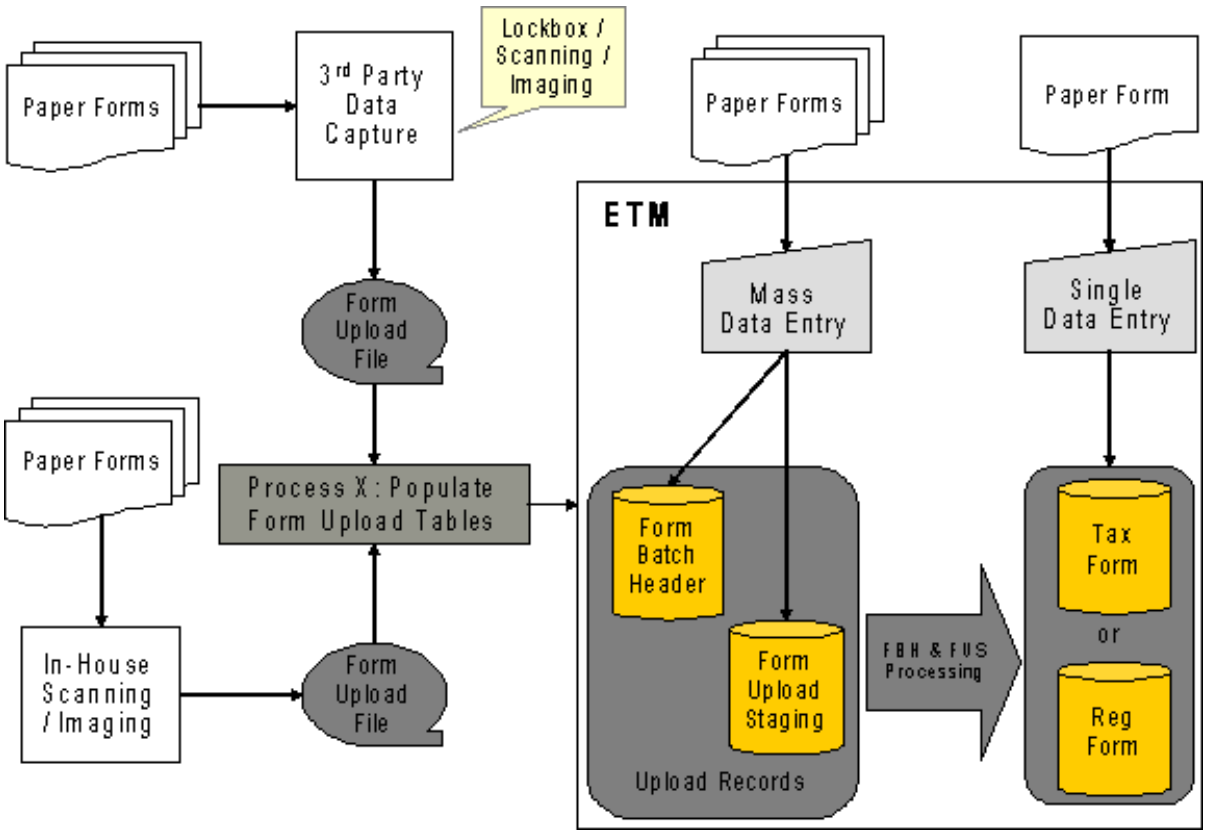
- Change reasons are only required if the status of the form is configured to enable versions.
- If the form type indicates that the Overall Change Reasons Applicability is **Required**, at least one form change reason should be supplied. On the other hand, if the form type indicates that the Overall Change Reasons Applicability is **Not Allowed**, form change reasons cannot be populated.
- If the form type indicates that the Line Change Reason Applicability is **Applicable**, a form change reason will be required for each line that has changed that is defined to have a form change reason. On the other hand, if Line Change Reason Applicability is **Not Applicable**, no line change reasons should be specified.
- Any form change reason that is populated on a form must be valid for the form's form type.

The Big Picture of Forms Upload

The following topics describe logic related to uploading batches of forms.

Forms Can Come From Different Sources

Forms are created in the system through a number of different ways. These methods of entering forms in the system are often referred to as 'channels'.



Most forms are uploaded in batch. This is a two-step process:

- Your implementation populates the form upload tables. That is referred to as 'Process X'.
- The base product processes the upload data and creates the corresponding tax forms and / or registration forms.

See *Designing Form Sources* and *Setting Up Form Sources* for details on how to configure your form sources.

Form Batch Header Groups Form Upload Records

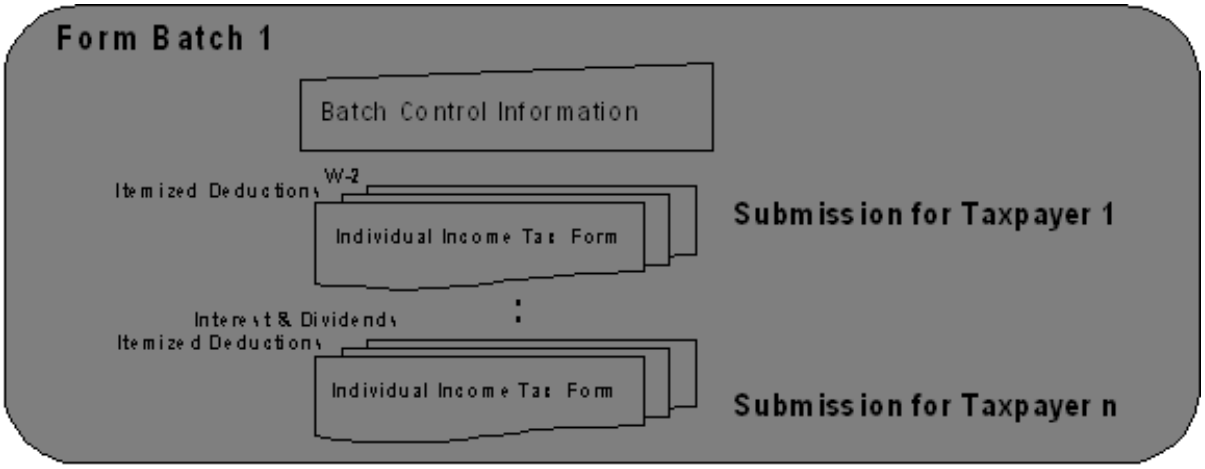
A form batch header is used to group form upload records together. It holds control information - e.g. number of form upload records, total payment amounts, etc.

A form batch header also controls the types of forms that can be put together in the same batch.

Refer *Designing Form Batch Headers* for details on how to configure form batch headers.

A Batch Can Group Forms Into Submissions

A batch can contain one or more submissions. A submission or a 'return' is a group of related forms that a taxpayer submits. It usually contains a main tax form and zero / one / more child forms. An example is an individual income submission where the income tax form is accompanied by a number of schedules and statements.



The product expects all documents supporting a single submission or 'return' to be included in a single form upload staging record that results in a single tax form or registration form, with sections representing the separate schedules and statements.

Form Upload Staging Holds Form Data

Actual tax / registration form data is uploaded via form upload staging record. A form upload staging record references a type, which determines how the upload data is mapped and stored in into the tax / registration form tables.

If tax / registration form creation is successful, the created form is linked to the form upload staging record.

Form Upload Staging Type Controls Everything

Each form upload staging record has a reference to its type. Form upload staging type controls how form upload data is stored into the product's tax form and registration form tables.

The bulk of your forms upload configuration will be spent configuring form upload staging types and defining the mapping of data from the different sources to your target form schemas in the system.

If your form sources each use a different record format / layout for interfacing a given form, a form upload staging type must be defined per unique combination of the destination form type and form source.

The mapping of the input form data into the target form schemas is done in the **Map Form Upload Data** algorithm plugged into the business object for the form upload staging type. The base product provides a form upload staging type BO **C1-FormUploadStagingType** that uses XSL transformation to map the raw XML data into the target tax / registration form schemas.

Note:

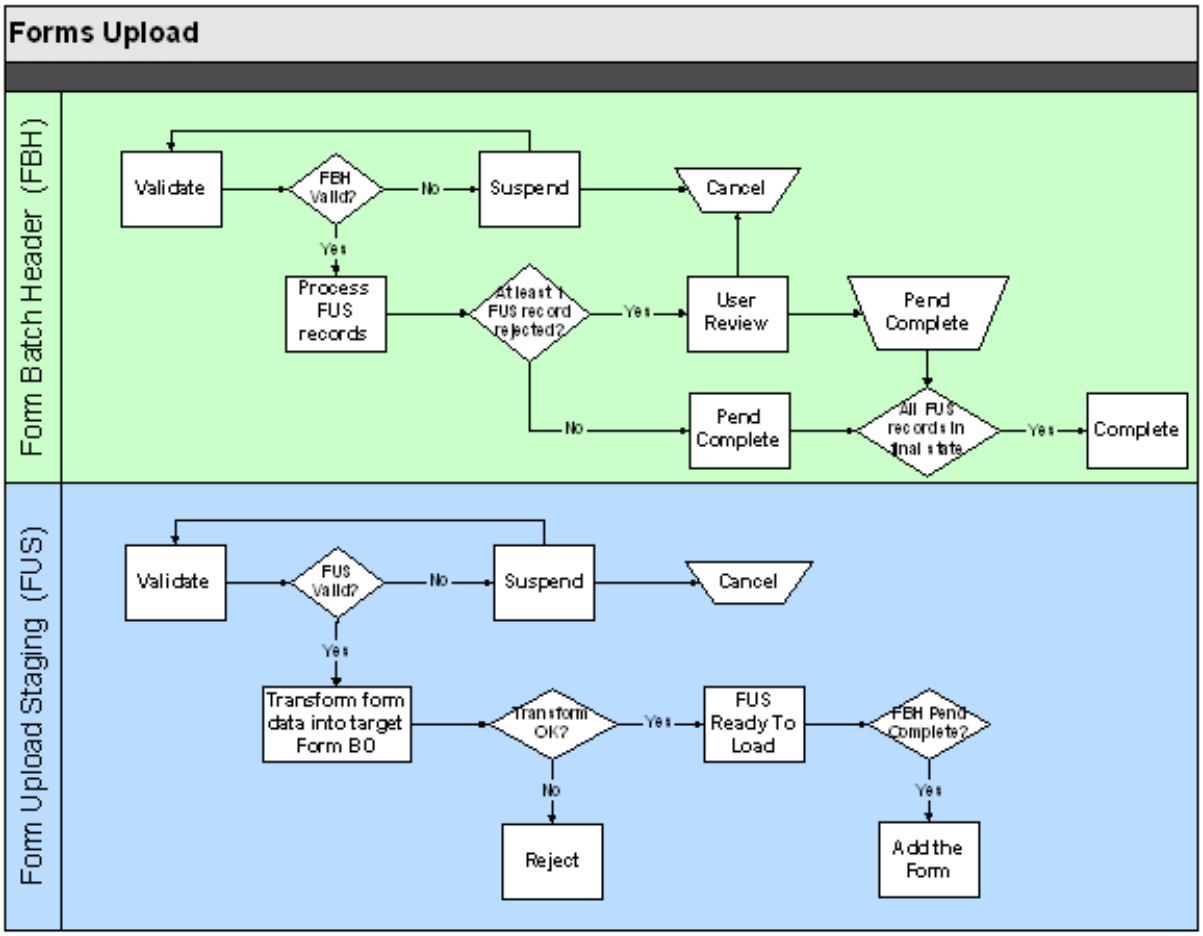
Base plug-ins. Click *here* to see the algorithm types available for this system event.

Refer to *Designing Form Upload Staging Types* and *Setting Up Form Upload Staging Type*s for details on how to configure your form upload staging types.

Forms Upload Process Flow

Process Flow for Form Batch Headers and Form Upload Staging Records

The following diagram illustrates the general process flows for form batch headers and form upload staging records.



Form upload staging records are not processed until the form batch headers they belong to have passed validation.

Any errors during the validation of a form batch header cause it to suspend for user review. The user is responsible for resolving the issues and re-validating the form batch header. At this point, the form upload staging records are still in their initial states. In some cases, the user may decide that the batch needs to be canceled, in which case the related form upload staging records are also canceled.

When the batch header passes validation, its form upload staging records can start processing. The batch sits in that processing state until either all staging records are in some final state - i.e. form added successfully, rejected or canceled.

Each form upload staging record goes through a number of processing steps before the corresponding tax form or registration form can be added. Any issues from these processing steps prevent the corresponding form from getting added.

- The form upload staging is validated. Any errors during form upload staging validation cause the form upload staging to get suspended for user review. The user is responsible for resolving the issues and re-validating the form upload record. In some cases, the user may decide that the form upload staging needs to be canceled. Note that at this point, the corresponding form is not created yet.
- When a form upload staging suspends, its batch header status stays the same - i.e. in the 'processing' state. The idea is that a user may be able to fix the issue with the staging record or decide to cancel it. Either action on the current staging record should not prevent other staging records in the batch from being processed.
- When the form upload staging passes validation, it goes through the important step of mapping the fields from the raw XML into the target Form business object schema.
- Any problems with this transformation are likely to be technical issues (e.g. malformed XML) that a user would not be able to fix. As a result, the form upload staging record gets rejected.

- On the other hand, if the transformation is successful, the form upload staging goes into a state where the tax form or registration form can be added / loaded.

If any of the form upload staging records is rejected, the batch header requires a user to review the batch and decide what to do next. A user may take either of these actions:

- Cancel the batch. This is likely when a majority of the records are rejected. Canceling the batch also cancels its form upload staging records. Note that at this point, there are no forms existing yet because the batch is in a cancelable state.
- Let the batch complete. This happens if only a few records in the batch got rejected.

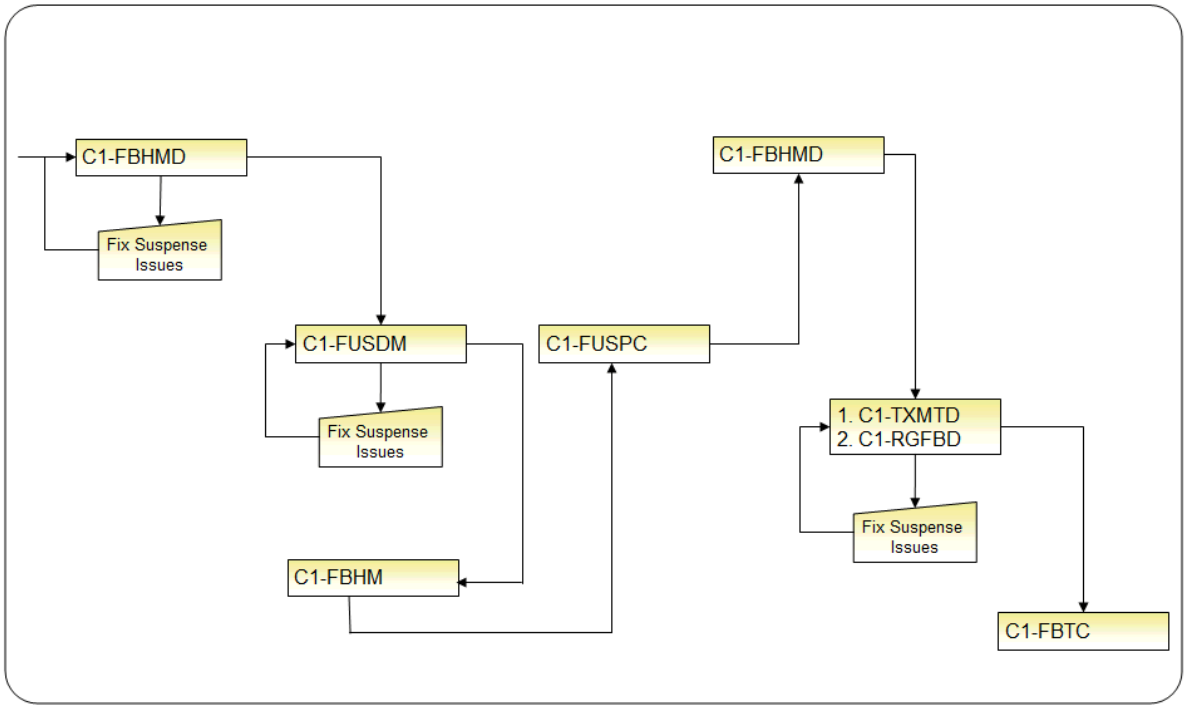
When all form upload staging records are either in **Ready for Load** state or are **Canceled**, the batch transitions to the **Ready to Complete** state. This interim state is one where batch cancellation is no longer possible. This ensures two things:

- That forms for that batch are added at the same time
- That once the forms are added, the batch cannot be canceled. Allowing batch cancellation when some forms are already added would require cleaning up the forms - i.e. canceling any pending / suspended / waiting forms get canceled and reversing any posted tax forms.

When the batch is in a **Ready to Complete** state, the forms in that batch can be added. When the forms are successfully added, their form upload staging records go to their final state, **Form Added**. Once all form upload staging records are in a final state, the batch is completed.

Forms Upload Batch Process Flow

The following diagram shows the forms upload process batch flow when using the base business objects for form batch header **C1-StandardFormBatchHeader** and form upload staging record **C1-FormUploadStaging**, and the base monitor background processes.



The processing of a form, from initial upload to posting is performed by the background processes as follows:

- **C1-FBHMD (Form Batch Header Deferred Monitor)**
  - Form batch header records transitioned from **Pending** to **Form Upload in Progress**.
  - Validates the form batch header records.

- **C1-FUSDM (Form Upload Staging Deferred)**
  - Form upload staging records transitioned from **Pending** to **Ready to Load**.
  - Validates the form upload staging records whose form batch header is in **Upload in Progress**.

  C1-FBHM (Form Batch Header Monitor)
  - Form batch header records transitioned from **Form Upload in Progress** to **Pending Complete**.
  - Updates the record counts on the form batch headers.
- **C1-FUSPC (Form Upload Staging Deferred Monitor)**
  - Form upload staging records transitioned from **Ready to Load** to **Form Added**.
  - Processes form upload staging records whose form batch header is **Pending Completion**.
- **C1-FBHMD (Form Batch Header Deferred Monitor)**
  - Form batch header records transitioned from **Pending Complete** to **Complete**.
  - Completes form batch header records.
- **C1-TXMTD (Tax Form Deferred Monitor)** and **C1-RGFBD (Registration Form Deferred Monitor)**
  - Tax form and registration form records are transitioned from **Pending** to **Posted, Suspended, Waiting for Information**.
  - Applying validation rules.
  - Create payments where applicable.
- **C1-FBTC (Balance Form Batch Tender Controls)**
  - Balances any tender control created for batch headers containing payments.
-

Note:

The sections above describe the batch processing and lifecycle transition of the form batch header and form upload staging records using the base business objects. Since this functionality is BO driven, it can be customized based on your specific implementation requirements.

Including Payments With Forms

Payments are uploaded with tax forms by specifying payments amounts on the corresponding form upload staging records. The actual sum of payment amounts from the form upload staging records within a batch are verified against the total payment amount specified on the form batch header.

The payment amount on the form upload staging record is simply copied over when the tax form is created. The actual payment may be created at different points in the tax form's lifecycle - depending on what's happening with the form.

- When the form posts, the obligation is already identified. Thus, the payment can be created for that obligation.
- When the form suspends, your implementation may want to create the financial transaction upfront and not wait for the suspense issue to be fixed. In this case, the payment can be created for a suspense obligation that is designated on the batch's tender source. When the form subsequently posts, the payment can be transferred to the correct obligation.
- When the form cancels, your implementation may want to create the payment anyway so that the financial transaction is still recorded in the system. In this case, the payment can be created for a suspense obligation that is designated on the batch's tender source. Manual follow-up is assumed in this case.

Refer to *Payments With a Tax Form* for more details on processing form payments.

Preparing To Upload Forms

The following sections describe the various considerations for preparing the system to upload forms.

Contents

Designing Form Sources

A form source is defined for each external source that interfaces forms into the system.

The following points describe a recommended design process:

- Define the categories for your form sources by updating the C1_FORM_CHANNEL_FLG lookup flag. Examples of these broad categories are: Lockbox, Third Party Data Capture, In-House Data Capture, etc.
- Determine the information about your external sources that the system needs to capture.
- Configure you tender sources. A unique tender source is usually associated to each form source.

The base product provides a form source business object **C1-FormSource** that maps all of the fields on the form source maintenance object.

If this BO's structure is sufficient for your implementation, you can use the BO as is. Your implementation can extend the BO's processing logic by adding / overriding algorithms and BO options.

If your implementation needs to capture additional information, you can: inherit from the base BO, copy the base BO or create your own BO.

Refer to existing help on configuration tools for more details on configuring business objects.

A Generic Form Upload Staging Business Object

The base product provides the form upload staging business object **C1-FormUploadStaging** that was designed to be generic such that your implementation can use it out of the box.

This BO has a raw XML (CLOB) field that is used to hold the actual form details. Staging the form details as raw XML provides the following benefits:

- A uniform way of interfacing form details:
- For different types of forms
- From various external sources that may upload data for a particular form type using different record structures/ layouts.
- External sources and Process X (i.e. the process that reads the input files and stores the form data into the form upload tables) do not need to know about the structure of the tax form and registration form business objects in the system.
- Form details can be uploaded into the system with minimal validation.
- As forms change in structure from year to year or as new form types are added, there is no need to create new form upload staging business objects. You implementation just needs to configure new form upload staging types (that reference the generic form upload staging BO), and define the mapping logic for the new forms.

The mapping logic configured on the form upload staging types takes care of transforming the data. This BO also has a transformed XML (CLOB) field that holds the transformed data. The value of this field is used when the tax / registration form is added.

Use **C1-FormUploadStaging** if the structure and lifecycle suits your implementation's form batch headers. As with any base BO, your implementation can extend the BO's processing logic by adding/overriding algorithms and BO options.

If your implementation needs to capture additional information, you can do any of the following:

- Inherit from the base BO - if the base BO lifecycle and processing logic works for you.
- Copy the base BO or create an entirely new BO - if your form upload staging records have a different lifecycle

Refer to existing help on configuration tools for more details on configuring business objects.

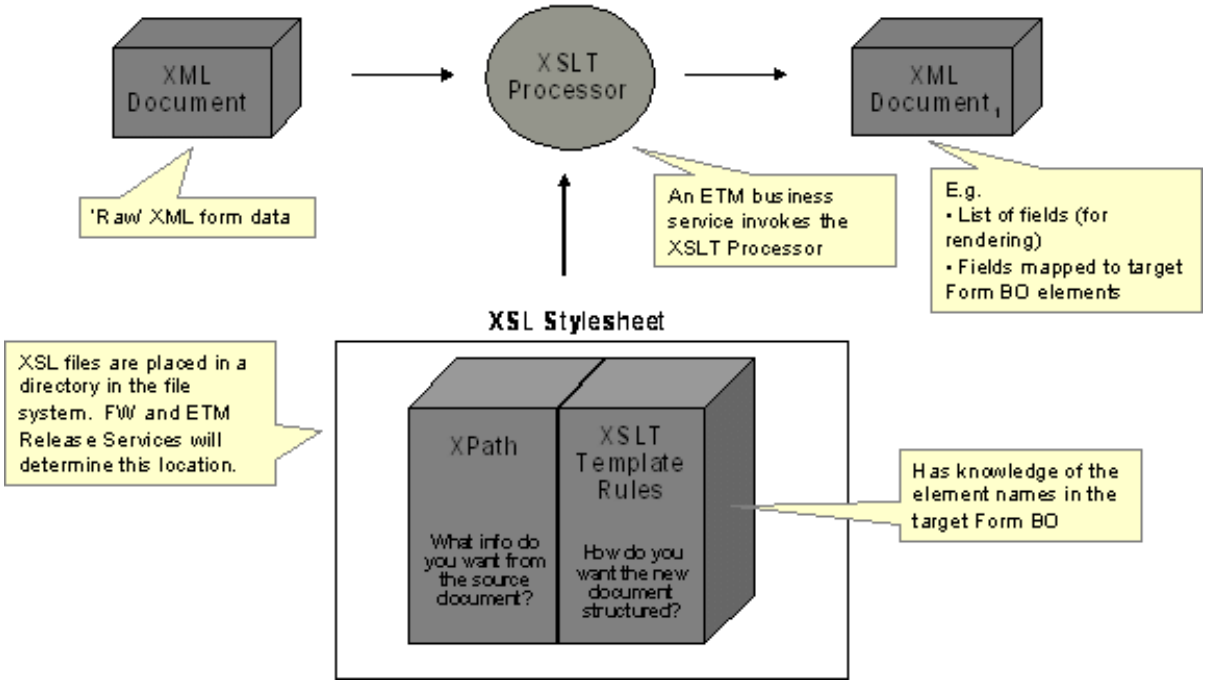
Designing The Mapping of Upload Data From The Different Sources

Given that the system expects the form data to come in as raw XML, your implementation needs to decide on the approach for mapping the raw XML data into the target form BO schemas. This decision usually entails looking at existing tools that are already available to your implementation.

Using XSL Transformation

The base product uses XSL transformation (XSLT) as an example for how to map the raw form upload data into the target form objects.

Since the input raw form upload data and the target form BO schemas are both in XML format, it makes sense to use XSLT. XSLT is a standard technique for transforming one XML document to another XML document. The key components in this approach are the XSLT Processor and the XSL style sheets.



XSL Transformation

The framework is already using an XSLT processor for transforming XMLs not only into other XMLs but also into HTMLs. A base business service calls an existing Java method for invoking that XSLT processor.

The base product provides a sample XSL file for mapping to the existing C1-SalesAndUseTaxForm business object - i.e. *C1-SalesAndUseTaxForm.xsl*.

Note:

The sample Sales & Use XSL logic assumes a very specific 'raw' XML data form that was not based on any single real-life sales & use form. This sample XSL is just meant to demonstrate this particular mapping approach.

Both base product and CM XSL files need to be stored in specific locations in the file system that ETM Release Services designate.

- Base files are stored in **@SPLEBASE@/splapp/resources/xsl/**.
- CM files are stored in **@SPLEBASE@/splapp/resources/xsl/cm/**.

XSL Transformation Is Not Required

Although base logic uses XSLT for transforming form upload data, your implementation is not required to use this approach. The base logic that invokes XSLT is in a 'Map Form Upload Data' plug-in on the Form Upload Staging Type BO. You can override the base logic by inactivating the base logic and adding your own. This is done as part of configuring your form upload staging types, which is described in the next section.

Designing Form Upload Staging Types

The following sections describe the various considerations for configuring your form upload staging types.

Contents

Form Upload Staging Type (Admin) Business Object

The base product provides the **C1-FormUploadStagingType** business object. This object maps all of the fields on the maintenance object plus additional fields (in the CLOB) for the suspense To Do Type and To Do Role. Since the form upload staging type does not have a soft status, **C1-FormUploadStagingType** does not have a lifecycle.

Use the base BO if the structure suits your form upload staging types. Otherwise, you can either subclass the BO or create an entirely different BO. Refer to existing help on configuration tools for more details on configuring business objects.

When opting to use the **C1-FormUploadStagingType** BO and your chosen mapping technique is not XSLT, you need to do the following:

- Inactivate the base algorithm **C1-FUST-XSLT (FUS Type - XSL Transformation)** by adding a BO option to **C1-FormUploadStagingType** with:
- BO option type = **Inactivate Algorithm**
- BO option value = **C1-FUST-XSLT**.
- Plug in your CM algorithm on the **Map Form Upload Data to Target Form** plug-in spot on the **C1-FormUploadStagingType** BO.

Note:

If your chosen approach is to not have mapping logic at all - i.e. Process X will populate the Transformed XML field in the staging record - you still need to plug in the CM algorithm, but the algorithm will do nothing but terminate.

C1-FormUploadStagingType also has BO options for automatic rendering of raw XML data on the UI.

- **Display HTML Generator XSL Stylesheet = @SPLEBASE@/splapp/resources/xsl/c1FormUploadStagingUIMapForDisplay.xsl**
- **Schema Extractor XSL Stylesheet** = **@SPLEBASE@/splapp/resources/xsl/c1FormUploadStagingSchemaForUIMap.xsl**
- **Maintenance HTML Generator XSL Stylesheet** = **@SPLEBASE@/splapp/resources/xsl/c1FormUploadStagingUIMapForInput.xsl**

Note:

Note. The generated maintenance HTML (third option) is just for editing raw XML data in existing form upload staging records. It is not used for adding form upload staging records online. A mass data entry solution is not included in this release.

If your implementation needs to render the raw XML data in a different way, you simply add option values for these three option types, using a higher sequence number. (For 'single' BO option types, the framework knows to get the highest sequence number.)

Defining Form Upload Staging Type Data

The following describes the steps in defining your form upload staging types.

- Determine the number of form upload staging types needed based on the number of your form types (that can be uploaded) and the sources that can interface those form types.
- Create a form upload staging type for each unique combination of form type and form source.
- Select the form upload staging type (admin) business object. See *Form Upload Staging Type (Admin) Business Object* for details configuring this BO.
- Specify the form upload staging type code (primary key)
- Put in a description of the form upload staging type
- Specify an **Active** status if you want to record to be effective immediately. Otherwise, specify an **Inactive** status and update the status later when the form upload staging type is ready.
- Specify the related transaction business object - the form upload staging BO. The base product provides a *generic form upload staging BO* for this purpose.
- Specify the form source.
- Specify the form type.
- If you are using XSLT to map raw XML form data to the target form BOs and you are using the base algorithm **C1-FUST-XSLT**, specify the XSL Transformation File Name. The file name must include the full path - e.g. **com/splwg/cm/domain/forms/formUploadStaging/xsl/salesAndUseForm.xsl**.
- If you selected the **C1-FormUploadStagingType** admin base BO, specify the suspense to do type and to do role.

Designing Form Batch Headers

The following points describe a recommended design process:

- Define the information that your form batch headers need to capture, e.g.:
- The valid form types that can be included in the batch - i.e. single vs. multiple
- Record counts
- Total amounts
- Examine the lifecycle of your form batch headers. Part of this effort is figuring out the points in the form batch header's lifecycle in which there is 'handshaking' with the form upload staging records' lifecycles.
- Design your form batch header processing algorithms.
- Identify issues that cause form batch processing to stop until a user can resolve the issues. Determine the appropriate notification action for such issues - e.g. To Dos
- Identify conditions that require a review of the form batch header. Determine the appropriate action for triggering such review - e.g. To Dos
- Identify conditions that cause form batch processing to complete. Determine the actions that need to take place at completion - e.g. final update of counters.
- Etc.

The base product provides the **C1-StandardFormBatchHeader** business object, which is designed to work for both batch uploads and mass data entry. It allows for multiple valid form types within the same batch. It also includes processing counters that capture the results of processing the included form upload staging records. This BO's lifecycle supports cancellation, validation, suspense, review and completion and is designed to work with the lifecycle of the form upload staging BO that is also provided in base - i.e. **C1-FormUploadStaging**.

Use **C1-StandardFormBatchHeader** if the structure and lifecycle suits your implementation's form batch headers. As with any base BO, your implementation can extend the BO's processing logic by adding/overriding algorithms and BO options.

If your implementation needs to capture additional information, you can do any of the following:

- Inherit from the base BO - if the base BO lifecycle and processing logic works for you.
- Copy the base BO or create an entirely new BO - if your form batch headers have a different lifecycle

Refer to existing help on configuration tools for more details on configuring business objects.

Setting Up Form Processing Options

The topics in this section discuss the various options for setting up the processing of forms.

Setting Up The System To Enable Forms Processing

This table describes all objects that must be defined as part of the forms processing setup process. It lists the order in which objects should be defined. It also identifies the other objects that are associated with or referenced by each setup object, thus providing a useful map of the relationships between objects in the system.

Seq	*Object*	*Description*	*Prerequisite*	*Associated With / Referenced By*	*Admin Submenu*
1	Form Type	Type of form	Obligation Type	Form Section, Form Line, Child Form Types, Valid Obligation Types, Tax Form, Registration Form, Form Upload Staging Type	Form Type
2	Form Section	Section of a form type	Form Type	Form Type, Form Line	Form Type
3	Form Line	A line in a section. Defines the type of information captured / displayed on the line	Form Section	Form Type, Form Section, Field, Lookup Field	Form Type
4	Form Change Reason	Predefined reasons for changes to information on the form		Form Type	Form Change Reason
5	Form Rule Group	A logical grouping of form rules		Form Type, Form Rule	Form Rule Group
6	Form Rule	Defines how information on the form is processed	Form Rule Group	Form Rule Group, Registration Form Exception, Tax Form Exception	Form Rule
7	Form Source	Predefined sources of forms	Tender Source	Tender Source, Form Upload Staging Type, Form Batch Header, Form Upload Staging, Tax Form, Registration Form	Form Source
8	Form Upload Staging Type	Type of form upload staging record	Form Source, Form Type	Form Source, Form Type, Form Batch Header, Form Upload Staging	Form Upload Staging Type

Setting Up Form Types

Form Types contain the rules that control how forms are processed. To set up a Form Type, open **Admin Menu** > **Form Type**.

Form Type Query

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

Form Type - Main

This portal appears when a form type has been selected from the Form Type Query portal.

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

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

If the record is in a state that has valid next states, buttons to transition to each appropriate next state are displayed.

Form Type

The Form Type zone contains display-only information about selected form type.

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

Where Used

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

Child Form Types

This zone displays the list of tax form types to which the current form type is a parent.

Section List

This zone lists the sections of the form type in sequential order. The Edit, Duplicate, and Delete actions are available. You can click the Add link in the zone's title bar to add a new section. You can click a form section hyperlink to go to the maintenance portal for that form section.

Line List

This zone accepts a broadcast form section and lists the section's lines in sequential order. The Edit, Duplicate, and Delete actions are available. You can click the Add link in the zone's title bar to add a new line. You can click a line hyperlink to go to the maintenance portal for that form line.

Form Type - Rules

To view the Form Type - Rules page, open **Admin Menu** > **Form Type** select a form type, and then navigate to the **Rules** tab.

This page displays the form type's rule groups in prime key order. A row is displayed for every rules linked to the form type rule group. The list includes the form rule information with a hyperlink to the rule's maintenance portal. You can add or edit the form type rule group by clicking the link in the title bar.

Form Type - Log

To view the Form Type - Log page, open **Admin Menu** > **Form Type** select a form type, and then navigate to the **Log** tab.

This page contains a standard *log zone*.

Setting Up Sections and Lines

Setting Up Form Sections

This portal appears when a section has been selected from the Section List on the Form Type portal or from the Current Form Type's Sections dashboard zone.

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

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Form Section

This zone contains display-only information about selected form section.

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

Form Section Line List

This zone accepts a broadcast form section and lists the section's lines in sequential order. The **Edit, Delete** and **Duplicate** actions are available. You can click the **Add** link in the zone's title bar to add a new line. You can click a line hyperlink to display the line details.

Setting Up Form Lines

This portal appears when a line has been selected from the Section Line List on the Form Type portal or the Form Section Portal.

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

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Form Line

This zone contains display-only information about selected form line.

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

Setting Up Form Change Reasons

Typically the tax authority will want to track the reason/s behind a form change, especially if the changes were made by a user. The user will be required to select from a dropdown of form change reasons whenever making a change that requires a reason. To set up a Form Change Reason, open **Admin Menu** > **Form Change Reason**.

The topics in this section describe the base-package zones that appear on the Form Change Reason portal.

Form Change Reason List

The Form Change Reason *List zone* lists every form change reason. The following functions are available:

- Click the *broadcast* icon to open other zones that contain more information about the adjacent form change reason.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each form change reason.
- Click the **Add** link in the zone's title bar to add a new form change reason.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Form Change Reason

The Form Change Reason zone contains display-only information about the form change reason. This zone appears when a form change reason has been broadcast from the Form Change Reason List zone or if this portal is opened via a drill down from another page.

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

Form Types

The Form Types List zone displays a list of form types that currently use the broadcast form change reason. The following functions are available:

- Click the **Add** link in the zone's title bar to link a new form type to the broadcast form change reason.
- Use the **Delete** action to delete a link between the broadcast form change reason and a form type.

Setting Up Form Sources

A form source is created for every external source that interfaces tax / registration forms into the system.

To set up a form source, select **Admin Menu** > **Form Source**.

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

Form Source List

The Form Source *List zone* lists every form source. The following functions are available.

- Click the *broadcast* icon to open other zones that contain more information about the adjacent form source.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each form source.
- Click the **Add** link in the zone's title bar to add a new form source.

Actions

This is a standard actions zone. The Edit, Delete and Duplicate actions are available.

Form Source

The Form Source zone contains display-only information about a form source. This zone appears when a form source has been broadcast from the Forms Source List or if this portal is opened via a drill down from another page.

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

Where Used

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

Setting Up Tender Sources

A *tender source* is created for each form source that interfaces payments with tax forms.

These tender sources will be used on the form batch headers' tender controls.

The tender source also specifies the suspense obligation to which payments should be applied if the tax form's obligation is not yet determined at the time of payment creation - e.g. when the form gets suspended or canceled.

To set up a form source, select **Admin Menu** > **Tender Source**. Refer to existing help on setting up tender sources.

Setting Up Form Upload Staging Types

A form upload staging type is created for a unique combination of form type and form source.

Use the Form Upload Staging Type transaction to view and maintain form upload staging types. To set up a form upload staging type, select **Admin Menu** > **Form Upload Staging Type** .

Form Upload Staging Type Query

Use the *query portal* to search for an existing form upload staging type. Once a form upload staging type is selected, you are brought to the maintenance portal to view and maintain the selected record.

You can click the **Add** link on this portal to add a form upload staging type.

Form Upload Staging Type Portal

This portal appears when a form upload staging type has been selected from the Form Upload Staging Type Query portal or if this portal is opened via drill down from another page.

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

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Form Upload Staging Type

The Form Upload Staging Type zone contains display-only information about the form upload staging type.

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

Where Used

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

Defining Penalty and Interest Options

Penalty and Interest (P&I) is the commonly used term to describe a wide range of penalty, interest and fee liabilities. These charges are usually imposed by legislation.

- A penalty is a liability imposed for noncompliance with tax law. Revenue authorities can file penalties against taxpayers for "failure to file," "failure to pay," "substantial understatement," and so on
- Interest compensates the government for cost of money over time
- Fees are imposed to reimburse revenue authorities for their cost of doing business. For example, revenue authorities may incur fees for sending certified mail or for handling defective checks and will pass those fees onto the taxpayers

P&I may be triggered in any of the following ways:

- Event driven, for example, reversing a payment due to non-sufficient funds will impose a fee. These types of transactions are simply one-off adjustments that are created by an appropriate algorithm when the event occurs
- User triggered, for example, auditors may impose an inadequate record-keeping penalty their discretion. The amount may be predefined or determined by the auditor. These types of transactions are created manually or may be created as part of a business process defined by your implementation.
- Ongoing, system-generated charges based on predefined configuration rules. The remainder of this chapter focuses on these types of charges.

The Big Picture of Credit Allocation

Before explaining penalty and interest logic, another important concept called credit allocation must be introduced. The following sections provide more information.

Credit Allocation

Credit allocation is the process of taking credits within an obligation and applying business rules to allocate them against tax, penalty, interest and fees.

Credit allocation sequences specify an order to apply the credit against liability types. This is usually very specific. For example, you could set up the credit allocation sequence to allocate credits in the following order:

1. Defective check fee
2. Failure to pay penalty
3. Interest
4. Failure to pay penalty
5. Tax
6. Outsource collection agency fee

They may also differ based on the type of credit. For example,

- Withholding credit may be allocated differently than a payment
- Payment received before the due date may be allocated differently than a late payment
- Individual payments may have unique priorities dictated by a court order or overridden by a user

The following sections provide more information about credit allocation.

Credit Allocation Is Dynamic

The term "dynamic" is used to describe credit allocation because the system does not store the matching of credits to debits in the database. Rather, the system applies the credit allocation rules real-time when a request for the detailed balance of an obligation is performed.

Debt Categories and their Priorities

In order to be able to apply credits to debits at the granular level shown above, each financial transaction whose amount is a debit (amount is greater than or equal to zero) must be assigned a debt category

- For adjustments that are debits, the debt category is defined *adjustment type*
- For bill segments, the debt category is defined on the *obligation type*
- For payments that are debits, for example "negative" payments used to refund money via direct deposit, the base product provides an FT freeze algorithm called *C1-CFTZ-RPDC* to plug into the Account Type that populates the debt category.

Debt categories may also be linked to credit financial transactions. The base algorithm that allocates credits to debits uses the debt category on credit financial transactions as a prioritization. See *Determine Balance Details Algorithm* for more information.

The debt category priority defines the order in which credits may be applied to debits.

- The default debt category priority is defined on the *obligation type*
- If a certain type of credit adjustment, for example a withholding credit dictates a different priority than the default, define the override debt category priority on the *adjustment type*.
- A *financial transaction* could reference a debt category priority directly for unusual situations where a payment may require a special debt category priority

Determine Balance Details Algorithm

The dynamic credit allocation is performed by an algorithm plugged into the **Determine Balance Details** plug-in spot on the *obligation type*. The determine balance details algorithm is used throughout the base P&I calculation algorithm but may also be called by other processes to get a balance broken out by assessment / debt category.

The following information highlights the logic in the base algorithm *C1-PS-DB-STD*.

The algorithm may receive specific FTs in which case it will only use those FTs. Otherwise, it retrieves the FTs for the obligation.

A special plug-in spot **Retrieve FT List** on *obligation type* is called to retrieve the FTs for an obligation.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

Note:

Debt Category / Assessment on Credits. This algorithm assumes that if a credit financial transaction references an assessment and/or a debt category that this information is used to prioritize the application of the credit, not restrict its application. This ensures that the results do not include one assessment (or debt category) with an overall debit balance and another assessment (or debt category) with an overall credit balance.

The algorithm processes credits in order from oldest to newest. For each credit found,

- If it references a debt category or assessment (group FT ID) or both, the credit is allocated first to debits with a matching debt category and / or assessment whose effective date is *on or before* the credit's effective date.
- If credit is still remaining, the credit is allocated to debits whose effective date is *on or before* the credit's effective date using the credit allocation rules for the appropriate *debt category priority* for this credit.

- If credit is still remaining, the credit is allocated to debits with a matching debt category and/or assessment whose effective date is *after* the credit's effective date.
- If credit is still remaining, the credit is allocated to debits whose effective date is *after* the credit's effective date using the credit allocation rules for the appropriate *debt category priority* for this credit.

The algorithm returns a detailed list of FTs including which credits were applied to which debits.

Note:

FT Details. The details of the FTs for the obligation and how the credits are allocated to the debits is not hard-coded but rather is defined in a data area supplied to the Determine Detailed Balance plug-in spot. The base algorithm uses the data area **C1-PI-MainFtInfo**. However, if your implementation requires different information to be supplied for each FT, you may introduce your own data area and an appropriate determine detailed balance algorithm to populate the details accordingly.

Credit Allocation Zones

The base product provides zones on control central, one on the account information tab and one on the taxpayer information tab, to show the results of credit allocation for the account or taxpayer's current balance.

Fastpath:

Refer to *Control Central - Account Information* for more details.

The Big Picture of Penalty and Interest

The following sections describe topics related to penalty and interest calculations.

P&I is Calculated for an Obligation's Assessments

The system creates, stores, and maintains tax liability *assessments*. These assessments may incur P&I over time. When more than one assessment exists for an obligation, the type of assessment may control P&I rates and rules. For example, if the taxpayer is being audited, harsher penalty rates and rules are used.

As described in *Credit Allocation* the base product determine detailed balance algorithm allocates credits for an obligation across assessments such that an obligation with multiple assessments would not incur P&I on one assessment while another assessment is in credit. As a result, the P&I calculation algorithm is invoked for an obligation and it always calculates penalty and interest for all assessments for the obligation.

P&I Rules for a P&I Control Define the Calculation

For each obligation type where ongoing, system generated penalty and interest rules apply, you must create a P&I control with its collection of P&I Rules that define the distinct penalty, interest charge or fee.

Each P&I rule controls the actual calculation and allows a user to configure information such as the basis of the charge, the rate used to apply the charge and other controls such as whether there is a maximum amount that the overall calculations may not exceed.

P&I rules to apply to an obligation or configuration related to existing rules may change over time, based on legislation changes or business rule changes. When this occurs, a new P&I control with its new set of rules must be created. The obligation type includes an effective dated link to the P&I controls that govern its P&I charges.

Fastpath:

Refer to *Apply P&I Rules for Each Time Period* for information about how P&I rule calculations are applied during the Calculate P&I process.

Fastpath:

Refer to *Designing Your P&I Control and P&I Rules* for information about designing your rules.

P&I Calculation Algorithm is the Engine

An algorithm plugged into the **P&I Calculation** plug-in spot on the *obligation type* is responsible for calculating / forecasting penalty and interest for an obligation.

Note:

Configuration Requirements. The base product P&I calculation algorithm relies on many other algorithms plugged into obligation type for it to work properly. In addition, it expects effective P&I controls with at least on P&I rule to exist for the obligation type. Refer to *Setting Up Penalty and Interest* for more information.

Calculation Overview

The following sections highlight the logic in the base algorithm *C1-PI-CALC*. It has been designed to call many other algorithm plug-in spots to perform key steps in the calculation. The goal is for your implementation to use the base product P&I calculation algorithm and implement your organizations specific rules by plugging in appropriate algorithms in the various plug-in spots called by the base algorithm. However if the logic in the base algorithm does not satisfy your business requirements, you may introduce your own, using this algorithm as a sample.

At a high level, the base algorithm follows these steps

- Determine whether or not this obligation is *eligible* for P&I calculations.
- For eligible obligations
- Determine *discreet time periods* and perform any other pre-processing needed for the calculations.
- For each time period, apply the P&I rules. Calculations are performed in memory.
- Post processing, including canceling and creating the P&I adjustments if the algorithm is called with a P&I Calculate / Update action.
- Update the obligation's calculate to date

The algorithm may be called with an action of **P&I Calculate/Update, P&I Standard Forecast** or **P&I Detailed Forecast**. When calling the algorithm with a "forecast" action, the calling program may provide specific FTs in which case it will only use those FTs. Otherwise, it retrieves the FTs for the obligation.

A special plug-in spot **Retrieve FT List** on *obligation type* is called to retrieve the FTs for an obligation.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

Financial Transaction Details

The list of the FTs for the obligation includes detail for each FT that is needed by the penalty and interest calculation. This list of details is not hard-coded but rather is defined in a data area supplied to the P&I Calculation plug-in spot and to the Retrieve FT List plug-in spot. The base algorithm uses the data area **C1-PI-MainFtInfo**.

Note:

If your implementation requires different information to be supplied for each FT, you may introduce your own data area.

Besides the details of the FTs passed back and forth to services that invoke this plug-in spot, the base P&I calculation algorithm uses data internally that must be passed to the various plug-in spots it invokes.

The base algorithms use the data area **C1-PI-InternalCalculationInfo**. The data area has the following information:

- A calculation periods collection. This includes start and end dates along with the P&I rules in effect for those dates.
- Existing FT collection. This includes all the FTs that exist for the obligation and is used to compare to the P&I charges being calculated in the current execution to determine if any changes are needed to historical periods.
- Running charges collection. This is the list of charges that are populated by the P&I rule algorithms that calculate the charges. The amounts in this collection are captured with detailed precision to minimize ongoing rounding discrepancies.
- Working financial transaction collection. This collection represents the new list of FTs that are a result of the current P&I calculation. When the service is invoked with an "update" action, this is the list that will be used when creating the actual FTs at the end of the process. If the service is invoked with a "forecast" action, this is the list of FTs that is used to pass out to the calling program. The definition of this list matches the definition of the list of FTs in the P&I Main FT Info data area. This list also includes a collection of detailed P&I calculation info that may be populated by P&I rule algorithms when invoked with a "forecast" action. Refer to *P&I Calculation Details* for more information.
- Waiver information collection. This list contains detailed information for any active waiver that exists for the obligation.

Note:

If your implementation requires additional information to be passed around to the various P&I calculation plug-in spots that support the internal P&I information, you may extend the base data area to define the additional elements needed. Alternatively, you may introduce your own data area. The appropriate decision for your implementation will depend on the type of additional information needed. For example, if your implementation requires some addition information about the taxpayer or the obligation to help determine rule eligibility, for example, then extending the base data area is the correct solution. However, if you require different or additional data within one of the existing groups in the base data area, it may be more correct to introduce a new custom data area.

Eligibility

There are some types of taxpayers that may be exempt from penalty and interest. Examples of such taxpayers include other government agencies and non-profit organizations.

The base product provides an *obligation type* plug-in spot **P&I Eligibility** where an implementation may plug in algorithms that check an obligation's eligibility for penalty and interest. The P&I calculation algorithm provided with the base product calls the P&I eligibility plug-ins prior to performing any penalty or interest calculations. If the algorithms indicate that the obligation is ineligible, no calculations are performed.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

Recalculate Every Time

Various events may cause the system to recalculate historic penalty and interest:

- Back-dated payment
- Waive existing penalty or interest
- Change to the obligation's override filing due date
- Canceling an assessment

For normal requests from the system to "bring P&I up to date", where historical calculations are not affected, the system still recalculates the P&I from the beginning every time.

One reason for this logic is to ensure that P&I is always accurate. It could be that something has changed in the system that does affect historical calculations and a request to recalculate historical P&I was not performed. The next time P&I runs, recalculating from the beginning ensures that the latest calculation reflects the correct charge.

Another important reason to recalculate P&I from the beginning every time P&I is brought up to date is to avoid rounding issues that may occur over time.

Consider the following example.

- A monthly penalty amount for a delinquent taxpayer is 34.3444
- If the system calculates, rounds and posts, each month ignoring the running total, the penalty after 2 months is
- Month 1 calculation: 34.3444, rounded to 34.34
- Month 2 calculation: 34.3444, rounded to 34.34
- Total: 68.68
- If instead the system keeps track of the running total and creates the adjustment based on the latest running total, the calculation is as follows:
- Month 1 calculation: 34.3444, rounded to 34.34
- Month 2 calculation: 34.3444 + 34.3444 = 68.6888 less amount already posted (34.34): 34.3488, rounded to 34.35
- Total: 68.69

Over time these rounding differences add up. To avoid that the system calculates from the beginning every time P&I is calculated and the running total is kept throughout the calculations.

Calculate For Discreet Time Periods

There are many discreet time periods for which P&I must be calculated:

- Every date for which a credit financial transaction exists (because the calculations are based on an outstanding balance, which is affected by each credit).
- Any effective dated change in the P&I Control linked to the obligation
- Any accrual day of the month. This is necessary if you have a monthly charge that should accrue on a given day of the month and the charge includes other charges in its calculation basis. For example, if penalty accrues on the first day of the month and includes interest in its calculation basis, you must stop and calculate interest and then the penalty on the accrual day of the month so that your calculation basis is accurate.
- Effective dates of any active waivers. If any assessments for the obligation have a waiver that is effective dated, the system should stop and calculated up to that date and then after that date for the waived amount to be accurate.
- More… Your implementation may identify other events in the system that affect P&I calculations. For example, if a bankruptcy is logged in the system as a case, P&I should be calculated up to the bankruptcy start date and then skipped until the bankruptcy end date

The base product P&I calculation algorithm relies on *P&I pre-processing* algorithms to build this list of dates. Once the list is built, the P&I calculation algorithm *applies P&I rules for each time period*.

Pre-Processing

When preparing to calculate penalty and interest for an obligation, the P&I calculation plug-in must gather some information that is needed throughout the processing. For example:

- The P&I controls and P&I rules in effect for the full calculation period.
- Information about waivers that are in effect for the obligation's assessments.
- Identifying the dates during the full P&I calculation period that affect the calculation such that the process should stop and calculate P&I up to that date.
- Determining whether some P&I rules are not applicable for an obligation or one of its assessments. For example, a failure to file penalty is only applicable for obligations where the taxpayer did not file the return on time; and perhaps only applies to the original return and not any amendments.
- Determining whether some P&I rules are only applicable during certain date ranges. For example, if a taxpayer has extended their filing date, perhaps the failure to file penalty is only applicable after the extended filing date but the failure to pay penalty is applicable starting from the original payment due date.

The system provides two plug-in spots to use for building up all this information.

- An *obligation type* plug-in spot **P&I pre-processing** is used to define dates and retrieve and configure information that is related to all P&I rules for the obligation type. For example, dates are built for all credit financial transactions for the obligation.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event.

- A P&I Rule plug-in spot **Rule Pre-processing** is used to define dates and retrieve and configure information that is specific to a given P&I rule. For example, a rule that is for the Failure to File penalty has a pre-processing algorithm to determine if the taxpayer failed to file on time.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event.

These algorithms populate information in the P&I Internal Calculation Info data area (**C1-PI-InternalCalculationInfo**) for use by subsequent algorithms in the process.

The following sections highlight additional details related to the responsibility of pre-processing algorithms.

Build P&I Controls and Call Rule Pre-processing Plug-ins

The base product provides an algorithm that should be the first pre-processing algorithm plugged in on your obligation types. It builds the initial date ranges for the P&I calculation, builds the list of P&I Controls and P&I Rules that are in effect for the full date range and, for each P&I rule, it calls the Rule pre-processing algorithms, if any exist.

For more information, refer to the algorithm type *C1-PI-PR-PIC*.

Marking Rules as Not Applicable Based on Periods

Although the initial pre-processing algorithm builds a list of all the P&I rules that are in effect for the full P&I calculation period for this obligation, there may be P&I rules that don't apply for the obligation or rules that should only apply for certain time periods. For example:

- The failure to file penalty should only apply if the taxpayer did not file the tax return by the due date.
- A collection fee that is based on the outstanding balance of the tax should only apply if the taxpayer was selected for overdue processing and should only apply to the time period on or after the creation date of the overdue process
- For a taxpayer that filed for bankruptcy, P&I Rules should not be applied during specific bankruptcy dates.

To mark P&I rules as not applicable for certain time periods (or for all the time periods built for the calculations) P&I pre-processing algorithms on the obligation type or rule pre-processing algorithms on the P&I rule should find the P&I rule in the calculation periods collection in the P&I - Internal Calculation Info data area and mark it as not applicable. Refer to the base product rule pre-processing algorithm Determine Failure to File Applicability *C1-PI-RA-DFF* for an example of this type of logic.

Marking Rules as Not Applicable for One or More Assessments

There may be P&I rules that only apply to a subset of the assessments for an obligation for one or more time periods. For example:

- The failure to file penalty may apply only to the original assessment and not to amendments
- Different failure to pay penalty rules may exist for the original assessment and for amendments
- For a taxpayer that filed for bankruptcy, P&I Rules should not be applied during specific bankruptcy dates.

To mark P&I rules as only applicable for certain assessments in a given time period, P&I pre-processing algorithms on the obligation type or rule pre-processing algorithms on the P&I rule should find the P&I rule in the calculation periods collection in the P&I - Internal Calculation Info data area and indicate the assessments that are not applicable. Refer to the base product rule pre-processing algorithm Determine Failure to File Applicability *C1-PI-RA-DFF* for an example of this type of logic.

Determine if Active Waivers Exist

The base product provides a plug-in *C1-PI-PR-WVR* to retrieve details about waivers that exist for any of the obligation's assessments. This information is used by any subsequent algorithm that may need information about the waivers.

Apply P&I Rules for Each Time Period

Once the discreet time periods for P&I calculation are determined, the base P&I calculation algorithm needs to determine the balance by debt category for the time period in question and call the P&I rule algorithms.

Balance By Debt Category During P&I Calculation

To calculate the balance by debt category, *Determine Balance Details* algorithm is used. However, because the P&I calculation algorithm is recalculating P&I from the beginning every time and needs the balance recalculated for each period, it needs to be explicit about which financial transactions are used in the balance calculation. For example:

- When calculating the balance as of a given date, credits effective on that date should be ignored because the system is trying to find the balance as of that date prior to applying the credit. However all debits on or before the given date should be considered.
- Existing P&I transactions should not be factored into the calculation, only the ones created by this P&I run.
- Existing *waiver* transactions should not be factored into the calculation. However, waiver transactions corresponding to the P&I transactions calculated in the current run must be considered.

To ensure that the appropriate financial transactions are used to calculate the balance for each time period, a special plug-in spot on the *obligation type* is used: **P&I Prepare Periodic Balance**. This plug-in is responsible for calling the **Determine Detailed Balance** plug-in passing the appropriate financial transactions. Note that this plug-in spot receives an indication as to whether it's the **Initial**, **Interim** or **Final** call to get the balance details. This information allows the plug-in to do extra steps at the beginning or at the end.

Note:

Base plug-in. Click *here* to see the detailed description for the base algorithm type available for this system event and for more information about the behavior of this plug-in spot.

Note:

Adjustment Category. The base plug-in for the Prepare Periodic Balance relies on the adjustment type category values of **Penalty and Interest** and **Waiver** to identify adjustments that should not be factored in during the **Initial** call to the algorithm. Care should be taken to only configure *adjustment types* with the Penalty and Interest category if they are adjustment types created through the P&I rules driven P&I calculation.

Once the balance is available for each time period, for each assessment linked to the obligation, the P&I calculation algorithm invokes the rule processing algorithm for each P&I rule in effect for the obligation type's P&I control during this time period.

P&I Rules Calculate the Charge

The actual calculation of each P&I rule's charge is done in the **Rule Processing** algorithm plugged into the business object for the P&I rule. Configuration that the rule processing algorithm needs to successfully calculate the charge for each period is often captured on the P&I rule when defining it.

All logic related to calculating the charge for the P&I rule, including determining if and how much of the charge is waived, must be done by a Rule Processing algorithm.

The following sections describe the responsibilities of algorithms of this type.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event.

Calculation Basis

In many cases the P&I charges are calculated as a percentage of outstanding debt, referred to as the calculation basis. The calculation basis may simply be the amount of unpaid tax or it may be the amount of unpaid tax plus other unpaid charges, such as unpaid penalty or unpaid interest.

The calculation basis can change over time based on changes to the legislation and recalculation of historic info should use the calculation basis in effect at the time of the charge. A change in calculation basis requires a new P&I control and a corresponding new set of rules to be created and linked to the obligation type for the correct effective date.

The base product P&I rule business objects allow the user setting up the P&I rule to configure one or more debt categories that are used as the calculation basis. The balance by debt category is calculated prior to the call to the P&I rules and is passed in using the working FT collection.

Adjustment / Debt Category Configuration

When the penalty or interest charge is posted to the system to affect the taxpayer's balance, an adjustment is used. The adjustment type is defined on the P&I rule.

Each penalty and interest charge is identified by a debt category and this debt category is important for P&I calculation. Because adjustment type references a debt category, the P&I rules supplied with the base product do not capture the debt category explicitly. They derive the penalty or interest rule's debt category from its adjustment type.

Note:

No Duplication of Debt Categories. The base algorithms for calculating P&I expect that no two P&I rules for a P&I control refer to the same debt category (via its adjustment type).

Calculating New Charges and Canceling Incorrect Charges

The Rule Processing algorithm is responsible for calculating the appropriate charge for the current time period. However, because P&I is recalculated from the beginning every time P&I is called for an obligation, it's possible for the calculated charge for this time period already exists in the database.

Once the new charge is calculated and added to the running FT collection, the algorithm should perform logic to align existing and running charges and update the working FT collection appropriately. The base product logic does the following:

- When the current calculation matches the existing P&I charge in the database, the existing charge is added to the working FT collection.
- When the current calculation is more than the existing P&I charge in the database, the existing charge is included in the working FT collection and a new entry for the incremental increase is added to the collection.
- When the current calculation is less than the existing P&I charge in the database, the existing charge is marked to cancel in the working FT collection and a new charge for the new amount is added to the collection. In other words, the base algorithms do not create negative P&I charges.

There are situations where a P&I charge is calculated and after that charge was created, something occurs to cause this P&I charge to no longer be applicable for the taxpayer or for the time period or for the assessment. For example, perhaps a backdated payment is entered and the calculation basis for this time period is now zero. In this case although no new calculation occurs, existing charges must still be marked to cancel. To do this, the algorithm should perform the logic to align existing and running charges whenever it detects that no charge is required. In this case, the running charges will have no entries for the time period / assessment so any existing charges get marked to cancel in the working FT collection.

Use Rates To Define the Penalty or Interest Rate

There are two ways that the actual percentage may be applied to the calculation basis: using a rate schedule or using a rate factor.

- Use a rate factor if the calculation is a simple percentage applied to the calculation basis. The rate factor can contain an effective dated list of percentage rates. Using this method, the appropriate rate factor is defined when configuring the P&I rule and the rule processing algorithm is responsible for retrieving the rate factor value and performing the calculation.
- Use a rate schedule if the calculation is more complicated and you would like to take advantage of the calculation logic built into the various types of rate components for a rate. For example, if your charge has a maximum amount that can be charged (for example, a penalty that is 2.5% of the outstanding tax up to a maximum 25% of the outstanding tax), this can be configured easily using the **Maximum** rate component type. Using this method requires the rule processing algorithm to set up all the amounts that the rate schedule requires to successfully apply the rate. For the current example, it must supply the outstanding tax amount that is the basis of calculation and the maximum amount allowed. Then it should call the rate application service. The P&I rule defines the rate schedule, rate quantity identifier (RQI) for passing in the calculation basis and any other information needed to supply to the rate application service. In this example, the Not to Exceed percentage and an RQI for passing to rates the calculated Not to Exceed amount.

Note:

Rate Schedule vs. Rate Factor. The simple percentage calculation may also be performed using rates. However, it requires an administrative user to configure a rate schedule, rate version and a rate component in addition to the rate factor. Designing the algorithm to apply the rate factor directly saves the need for extraneous rates configuration.

Waiver Processing

As charges are being calculated for each period, rule processing algorithms must also determine whether or not these charges should be waived and enter appropriate entries in the working FT collection to represent the waived amounts.

The base product provides a separate algorithm for each *supported waiver type* in the system that should be plugged into any P&I rule whose charge may be waived. Each algorithm does the following:

- Determine if there is a waiver in effect for the current debt category / assessment. This information is in the Waiver information collection in the P&I - Internal Calculation Info data area assuming that the pre-processing algorithm to *Determine if Active Waivers Exist* is plugged in.
- The algorithm then does the appropriate logic to determine whether the amount should be waived based on the logic for the type of waiver.
- If the amount should be waived, an entry is added to the working FT collection. This ensures that subsequent balance calculations for this debt category do not include the amount that is waived.

No adjustments are created or canceled at this time. Post processing algorithms are responsible for that logic.

Sample P&I Rule Processing Logic

The following points highlight typical processing for a P&I rule processing algorithm that calculates a penalty or interest charge.

- The algorithm should first check whether the charge is applicable for the current period and for the current assessment. This information is populated in the base internalPenaltyAndInterestDataArea by a P&I pre-processing algorithm that may determine that this charge is only applicable for certain time periods and / or for certain assessments. If it's not applicable, the algorithm checks to see if existing charges need to be cleaned up using the 'align existing and running charges' logic (see below).
- If the charge is applicable, then the algorithm should apply the charge as per the business rules.
- If the P&I calculation is called with the P&I Detailed Forecast action, the algorithm should populate appropriate *calculation details* in the *calculation info* collection.

- The algorithm should include a step to *Align Existing and Running Charges*, which should compare the existing charges in the database (from the previous P&I update) to the current running P&I collection.
- If the total of the current running charges is more than the existing charges a new charge should be added to the working FT collection for the difference.
- If the total of the current running charges is less than the existing charges, the base recommendation is for the algorithm to mark the appropriate existing charges as "canceled" in the working FT collection by populating the Cancel Reason from the P&I Rule and to add a new entry for the new smaller amount.
- If the P&I calculation is called with the P&I Detailed Forecast action, the calculation details from the running charges should be added to the working FT collection.

Note:

No updates. The rules processing algorithms provided in the base product do not do any updates to the database for new charges or cancellation of existing charges. These algorithms simply update the internal P&I information with new charges calculated and indicating any charge that should be canceled. Post processing algorithms are responsible for creating / cancelling financial transactions if Calculate P&I is called with the Calculate / Update action.

Note:

The base algorithms rely on data provided in the base P&I Main FT Info data area (C1-PI-MainFtInfo) and the base P&I internal calculation info data area (C1-PI-InternalCalculationInfo).

Post Processing

The system provides an *obligation type* plug-in spot **P&I post-processing** to perform any steps that need to occur at the end of all the P&I calculations.

In the base P&I calculations, post processing algorithms are used to do the actual creation and cancellations of penalty and interest adjustment and waiver adjustments in the system as determined by the rule processing algorithms. This is only applicable when the P&I algorithm is called with a **P&I Calculate / Update** action.

Note:

Base plug-in. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

Final Updates

When penalty and interest is updated for an obligation, it is updated with a Calculation Through Date. This information is captured so that users reviewing the detailed balance for an obligation knows how recently the P&I calculation has occurred.

The balance details are calculated one more time to produce the final results for output.

Forecasting Penalty and Interest

Your implementation may include business processes that require forecasting of an obligation's balance to a current or future date without posting the P&I transactions:

- Sending a bill may include amount to pay if late, for example:
- If you pay by January 1, please pay $2000
- If you pay by January 15, please pay $2020
- Etc,
- Sending a collection notice may include "please pay X amount by Y date" where the X amount is the forecasted balance on that date, which P&I included.
- If a payment is received at an account or taxpayer level where the taxpayer has several unpaid obligations, the payment algorithm that determines how to distribute the amount across obligations. When determining how much

to direct to each obligation, the algorithm should forecast P&I to the payment event's effective date so that an accurate picture of the balance of each obligation is known.

- *Proposing scheduled payments* for a pay plan should include logic to forecast P&I to the future so that the scheduled payments cover P&I charges that will continue to accrue

The P&I calculation algorithm may be called to forecast. When forecasting, the algorithm perform all the same calculations, but does not store or cancel any adjustments to affect the obligation's balance.

Fastpath:

The credit allocation zones on control central allow a user to forecast P&I for a current or future date. Refer to *Control Central - Account Information* for more details.

A service that calls P&I calculation with a "forecast" action may optionally provide financial transactions to use. If financial transactions are not provided, the algorithm retrieves the FTs currently linked to the obligation and forecasts P&I to the input date. If FTs are provided, the algorithm uses them in the P&I calculations. Supplying FTs allows a calling program to forecast the P&I with a "what if" scenario, for example "what if the taxpayer makes a payment on date X?". This is useful when proposing scheduled payments for a payment plan.

P&I Calculation Details

The product provides a page where a user can view the details of the penalty and interest calculation for a given obligation. However, because the calculation of each penalty or interest transaction is governed by the algorithm linked the P&I rule, the algorithm is responsible for supplying the details shown on this page. this section provides more information on how to configure the system to provide P&I details.

Fastpath:

Refer to *Detailed P&I View* for more information about the page and its behavior.

Details Captured for Each P&I Transaction

The working FT section of the internal P&I data area used during the penalty and interest calculation include a collection of one or more *calculation info* strings that can be populated with the details related to the calculation of the charge. This information is passed out to the calling program using the main P&I data area.

The P&I Rules Define the Details

Because each P&I charge is calculated by an algorithm, it's the algorithm's responsibility to provide the details of the calculation.

The base product algorithms populate the detail based on their specific calculations. Click *here* to see the algorithm types provided by the base product and view their description details for more information.

If your implementation defines custom P&I rule algorithm to support your specific calculations and your users may wish to view the details of the P&I calculation using the Detailed P&I View page, you should populate one or more *calculation info* strings in the P&I data areas appropriately.

Details Provided During Forecast Only

The base product does not store the calculation details with the adjustments created for penalty and interest. The details are only provided by the base product algorithms when the Calculate P&I service is called with the **P&I Detailed Forecast** action.

P&I and Cash Accounting

It is common for tax authority to practice cash accounting for certain situations. For example, perhaps your implementation is a state revenue agency collects sales tax revenue on behalf of counties in your state. You may only distribute the revenue to the counties after the payment is made and not at the time of posting the assessment.

Fastpath:

Refer to *Payables Cash Accounting* for general information about cash accounting functionality.

When dynamic credit allocation is practiced, where amounts are directed specifically to tax, penalty, interest and fees in a specific order, the allocation of credits to the various debt categories must be performed before any updates to the general ledger can be made as a result of cash accounting.

The base product provides an *obligation type* plug-in spot **Cash Accounting True Up** where an implementation may plug-in the algorithm that makes all necessary adjustments to the general ledger to transfer amounts from "holding" general ledger accounts to true payable accounts.

This algorithm is executed by the Calculate P&I service provided with the product when invoked with an "update" action. The service determines if the obligation type supports the P&I plug-in and if so, executes it. If the obligation type has a Cash Accounting True Up algorithm, it is then called, regardless of whether the obligation type supports P&I.

It means that any update to the system that impacts a taxpayer's financials where either P&I or Cash Accounting may be affected, the Calculate P&I service should be called. The business service is **C1-CalculatePenaltyAndInterest**.

Note:

Base plug-ins. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

When Is P&I Updated?

There is no hard coded logic in the system to invoke the penalty and interest calculation. All requests to bring penalty and interest up to date are executed using plug-in logic. The base product provides logic to forecast P&I or bring P&I up to date when certain events occur. Your implementation may introduce new events that should cause P&I to be recalculated.

The following events in the system cause P&I to be brought up to date

- When a *tax form* posts or is reversed, transferred or adjusted, using an appropriate Form Rule. Basically any state transition of a form that causes adjustments to be created should include a rule to bring P&I up to date.
- When a payment is frozen or canceled, assuming that an appropriate plug-in is entered on the *account type*
- When the *effective date* of a frozen payment is changed
- If your *obligation* is governed by the **C1-FilingPeriodObligation** business object, a post processing plug-in recalculates P&I if the override due date changes.
- When a *waiver* is activated or canceled
- An *overdue event* algorithm to calculate P&I is provided

A user may also request that P&I is brought up to date using the Credit Allocation zones on Control Central on both the *Account Information* portal and the *Taxpayer Information* portal.

C1-CALPI - Bring P&I Up to Date

Besides the events in the system that may cause a recalculation of P&I, the system also provides a background process *C1-CALPI* to periodically bring P&I up to date for all obligations eligible for P&I processing.

The background process looks for all non-canceled non-closed obligations and invokes the calculate P&I service, which then invokes the appropriate P&I calculation algorithm for the obligation, if one is plugged in on the obligation type.

Adjustments and Updating P&I

The base product does not provide a plug-in to bring P&I up to date when an adjustment is frozen or canceled.

- Certain adjustments are created / canceled from the P&I calculation, for example the penalty and interest charges and waiver charges. For these types of adjustments, penalty and interest recalculation should not be invoked.
- For other adjustments where P&I should be brought up to date, there are many business scenarios where several adjustments are created for various charges. For example, when a tax form is processed, adjustments may be created for the tax assessment due and for the withholding credit and for refundable tax credits. The system should wait until all the adjustments are created before bringing P&I up to date, otherwise a lot of unnecessary calculations will occur.

Rather than building "update penalty and interest" logic into a plug-in for the adjustment, the expectation is that a business process that creates adjustments that affect P&I should include a step to invoke P&I when the adjustments are created.

If your implementation has manually created/canceled adjustments that should affect penalty and interest, you should plug-in Adjustment Freeze and Adjustment Cancel plug-ins to bring P&I up to date.

Also note that when an assessment adjustment is canceled, all related penalty and interest adjustments should be canceled as well. The P&I calculation does not cater for cleaning up penalty and interest for canceled assessments.

Note:

Special service. When cancelling an assessment adjustment, the business service **C1-CancelAssessmentAdj** should be used. This will clean up penalty and interest adjustments for the canceled assessment.

Waivers

Waivers provide the ability to forgive or waive certain penalties, interest or fees. Sometimes waivers are referred to as abatements. You can waive a full amount or a partial amount (flat rates or percentages) or waive charges during a certain time period.

What is Waived?

A waiver is created for a specific assessment and a specific debt category for that assessment. For example, interest for the obligation's original assessment may be waived or the failure to pay penalty for the amendment may be waived.

Supported Types of Waivers

The base product provides business objects and appropriate algorithms out of the box that support the following types of waivers:

- One-time waivers. Waivers of this type are used to waive charges for a given assessment / debt category up to a specific amount.
- Effective dated waivers. Waivers of this type are used to waive charges for a given assessment / debt category that occur on or after a given start date. Waiver of this type may optionally specify an end date to waive charges for a specific period.
- Ongoing waivers. Waivers of this type are used to waive all charges for a given assessment / debt category.

Waivers vs. Exemption

A waiver is treated differently from an exemption in the base product algorithms. The assumption is that when an obligation is exempt from penalty and interest, calculations should not even be performed. Refer to *Eligibility* for more information.

For a waiver, the base product calculates the penalty or interest charge as normal and then creates appropriate waiver adjustments to offset the waived amounts. This allows an implementation to keep track of the amount of penalty and interest that has been waived.

Waivers and P&I Calculation

In order for waivers to affect P&I calculation, appropriate algorithms must be plugged in. The following points highlight the algorithms that are required to support waiver functionality:

- Information about existing waivers should be retrieved by a P&I pre-processing algorithm for use by the P&I rule processing algorithms. Refer to *Determine if Active Waivers Exist* for more information.
- If your implementation supports effective date waivers, the P&I pre-processing algorithm **C1-PI-PR-WDT** Adjust Calculation Periods for Effective Waivers should be plugged into any obligation type that supports P&I processing with waivers.
- Every P&I rule that calculates a charge that may be waived should include appropriate *rule processing waiver* algorithms plugged into the P&I Rule BO to adjust calculated charges in the internal P&I information data area based on existing waivers. Algorithms are provided to support one-time waivers, ongoing waivers and effective dated waivers. The base product P&I rule BOs are configured with the waiver algorithms already plugged in.
- *P&I post processing* algorithms **C1-PI-PS-WV1** Process One-time Waivers, **C1-PI-PS-WV2** Process Ongoing Waivers and **C1-PI-PS-WV3** Process Effective Dated Waivers are provide to create or cancel waiver financial transactions when calling P&I with an "update" action. These algorithms should be plugged into any obligation type that supports P&I processing for waivers.

Note:

No excess waiver. The base product algorithms assume that when a waived penalty or interest charge is later modified to cause the original amount to be less than the waiver, the waiver amount must also be adjusted accordingly. A waiver credit should never be in excess of the debt it's waiving.

Designing Your Waiver Options

The base product provides admin and transaction business object pairs for one-time waivers (**C1-WaiverTypeOneTime** and **C1-OneTimeWaiver**), ongoing waivers (**C1-WaiverTypeOngoing** and **C1-OngoingWaiver**) and effective dated waivers (**C1-WaiverTypeEffectiveDated** and **C1-EffectiveDatedWaiver**). Your implementation can add additional business rules to these BOs as required. If your implementation has waiver rules that aren't satisfied by one of the above business objects, you may create your own using the above as samples.

Designing Your P&I Control and P&I Rules

The base product provides a BO for P&I Control **C1-StandardPIControl**. Your implementation can add additional business rules to this BO as required. If your implementation has radically different requirements for your P&I controls, you can create a different business objects with their own business rules.

The base product supplies two BOs for P&I Rules.

- **C1-MonthlyChargePIRule** - this BO includes an algorithm that calculates the monthly charge with a Not to Exceed limit, using a rate schedule.
- **C1-SimpleCalculationPIRule** - this BO includes an algorithm that calculates a simple charge applying a rate factor to the outstanding calculation basis amount.

Your implementation can define add additional business rules to these BOs as required. If your implementation's P&I rules are not satisfied by one of the above business objects, you may create your own using the above as samples. The following points highlight the important configuration for this business object:

- Develop the appropriate rule processing algorithm to calculate the charge correctly.
- If *waivers* are applicable for the P&I rule's debt category, be sure to plug in appropriate rule processing algorithms to handle waiving the P&I rule's charge
- For any configuration required by the rule processing algorithm, consider whether it should be defined when configuring the P&I rule by a business user. If so, include the appropriate elements in the P&I rule business object's schema.
- Determine whether any P&I Rule *Pre-processing* algorithms are applicable for this type of rule.

Setting Up Penalty and Interest Options

The topics in this section describe how to set up the system to enable penalty and interest calculations.

Setting Up Account Types

In order for penalty and interest to be brought up to date when a payment is frozen or canceled, configure the account types with the following algorithms:

- System Event: **Payment Freeze**, Algorithm **C1-PI-PAYFRZ**
- System Event: **Payment Cancellation**, Algorithm **C1-PI-PAYCAN**

Setting Up Debt Categories

Define *debt categories* for the following:

- One for each separate penalty or interest charge that is calculated by a system P&I rule.
- One for any other type of debt that is not calculated by P&I calculations, but is used in the calculation basis for one of the P&I rules.
- One for any other category of debts. For example, overpayments. When an obligation is overpaid, at some point the *overpayment* amount is reduced by one or more methods, such as minimum amount write down, carry forward to a future obligation, refund, etc. The adjustments created to reduce an overpayment are debits and therefore require a debt category. The suggestion is to create a special "Overpayment" debt category for these types of debts.

Setting Up Debt Category Priorities

Define appropriate as *debt category priorities* required by your implementation's business rules. Refer to *Debt Categories and their Priorities* for more information.

Setting Up Adjustment Types

Adjustment types are needed for posting P&I charges.

- A separate adjustment type must be created for each debt category that is part of the P&I calculation.
- Each adjustment type created should use the **Penalty & Interest** adjustment type category.

Adjustment types are needed for posting waivers.

- A separate adjustment types must be created for each debt category whose P&I charges may get waived.
- Each adjustment type should refer to the **Waiver** adjustment type category.
- Each adjustment type should define the **C1-WAIVR** characteristic type as a valid template characteristic. This is used to reference the waiver that caused the adjustment to be created.

An adjustment type is needed for transferring cash accounting amounts from a "holding" general ledger account to the true "payable" account.

For any adjustment that is a manual penalty and should cause P&I to be recalculated, configure and adjustment freeze algorithm to bring P&I up to date. Note that the base product does not supply such an algorithm.

Setting Up Adjustment Cancel Reasons

Define appropriate *adjustment cancel reasons* to use when cancelling P&I adjustments and based on recalculation and cancel reasons to use when cancelling waiver adjustments.

Setting Up Rate Factors

Create appropriate *rate factors* that define the percentage to use when calculating your various P&I charges.

- If you are defining P&I rules using the base BO **C1-SimpleCalculationPIRule**, the P&I rule requires a rate factor to use when applying the charge.
- If you are defining P&I rules using the base BO **C1-MonthlyChargePIRule**, the P&I rule expects to use a Rate Schedule. You may choose to define rate factors for configuring the charges used in the specific rate components defined in this rate schedule.

Setting Up Rate Quantity Indicators

Create appropriate *RQIs* required for your P&I rules. If you are defining P&I rules using the base BO **C1-MonthlyChargePIRule**, you need an RQI for the following information to pass into rate application:

- The calculation basis amount
- The not to exceed amount

Setting Up Rate Schedules

Create appropriate *rate schedules* and their components to calculate the charges appropriately.

If you are defining P&I rules using the base BO **C1-MonthlyChargePIRule**, you need a rate schedule with rate components that calculate the charge by applying an appropriate percentage to the calculation basis passed in using the RQI defined above. If a Not to Exceed amount is applicable, there should also be a maximum rate component that adjusts the calculated charge based on the Not to Exceed amount.

The following are some additional notes related to the setup of this rate:

- The charges are not expected to prorate for short or long periods. To accomplish this, an appropriate frequency must be configured for the rate schedule. Because the P&I rule is expecting to apply monthly charges, the frequency may be set to have 1 period annually and 365 days for the minimum and maximum offset. Refer to the demonstration data for an example.
- The individual rate components should all be set as follows:
- FCPO is checked
- The Result Type is **Charge**
- Rounding Precision is set to the maximum allowed to ensure that the maximum calculation precision is returned.
- Create Bill Line is checked
- FCPO Retention Rule is set to **Retain amount on bill line**
- To support *calculation details*, appropriate description on bill values should be populated and the Print flag checked.

Setting Up P&I Control

A **P&I Control** is created to hold all the rules that work together to calculate automated / ongoing penalty and interest. The P&I control includes a list of **P&I Rules** that make up the individual calculations.

To set up P&I Control, select **Admin Menu** > **P&I Control**.

The topics in this section describe the base-package zones that appear on the P&I Control portal.

P&I Control List

The P&I Control List zone lists every P&I Control. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent P&I Control.
- Click the **Add** link in the zone's title bar to add a new P&I Control.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

P&I Control

The P&I Control zone contains display-only information about a P&I Control. This zone appears when a P&I Control has been broadcast from the P&I Control List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

P&I Control Obligation Types

This zone lists obligation types currently linked to the broadcast P&I control.

P&I Rules

This zone lists the P&I rules associated with the broadcast P&I control. Click on a P&I rule in the list to view its details in the *P&I rule portal*.

If the P&I control is in **pending** status, Edit and Delete actions are available for each P&I rule. In addition you may add another P&I rule using the **Add** link in the zone's title bar.

P&I Control Log

This is a standard *log zone*.

Setting Up P&I Rules

This portal is provided to view and maintain details of a given P&I rule. This portal is not available from a menu. You navigate to this portal by drilling into a specific P&I rule from the *P&I control* portal.

The topics in this section describe the base-package zones that appear on the P&I Rule portal.

Actions

This is a standard actions zone. The **Edit** and **Delete** actions are available.

P&I Rule

The P&I Rule zone contains display-only information about a P&I rule.

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

Where Used

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

Setting Up Obligation Types

Each obligation type where penalty and interest charges are applicable must be configured appropriately:

- Define the default *debt category priority*.

- It should include appropriate P&I controls that define the P&I rules that are in effect
- It should include appropriate P&I calculation algorithms. The following provides a list of the plug-in spots that must are required for P&I calculations. Unless noted, base algorithms are provided. If the functionality provided in the base algorithms does not satisfy your implementation's business rules, you may define your own:
- P&I Calculation
- Determine Balance Details
- Retrieve FT Details
- P&I Prepare Periodic Balance
- P&I Pre-processing
- P&I Post Processing - algorithms are provided for all algorithm types except for the *cash accounting* algorithm type. That one needs to be configured with the appropriate adjustment type.
- In addition to the required algorithms above, if your implementation's business rules include P&I Eligibility requirements, provide appropriate algorithms.

Setting Up Waiver Types

To open the Waiver Type zone, select **Admin Menu** > **Waiver Type**.

Waiver Type List

The Waiver Type List zone lists every waiver type. The following functions are available:
- Click a *broadcast* button to open other zones that contain more information about the adjacent waiver type.
- Click the **Add** link in the zone's title bar to add a new waiver type.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Waiver Type

The Waiver Type zone contains display-only information about a Waiver Type. This zone appears when a Waiver Type has been broadcast from the Waiver Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

Waiver Type Log

This is a standard *log zone*.

Defining Overpayment Processing Options

An overpayment occurs when the payment(s) exceeds the liability in an obligation, causing the obligation's balance to become a credit. Not every obligation that is in credit is considered to be an overpayment. This will typically depend on the tax type and the authority's rules.

The Big Picture of Overpayments

For return-based taxes, such as individual income tax, a credit balance will be considered an overpayment only once a return is filed. Any payments or credits received in advance of a tax return are not considered overpayments until

the tax form is processed, and an assessment is created. If a tax return is not filed, the obligation will be in a credit balance until the tax authority reviews it.

For billing-based taxes, different rules may apply to determine if an obligation is overpaid.

The following are common scenarios that would result in an overpayment:
- A majority of taxpayers receive refunds for their individual income tax filing because employer withholding tends to exceed the probable liability.
- Recalculation of tax, penalty, interest or fees that results in a reduced liability.

Not all tax types have overpayments. Assessments for sales tax and withholding tax are seldom overpaid because the taxpayer pays for an activity that has already happened. If an overpayment results, it is usually due to a math error on the form.

An Overpayment Process Can be Created Automatically or Manually

An overpayment process can be created in a variety of ways:
- As a result of posting a return. A form rule related to the posting system event could be used to create an overpayment process if the form reports a refund.
- An account monitor rule can be designed to create an overpayment process when the obligation's balance is a credit and a return has already been filed. The base-package provides the algorithm **Initiate Overpayment Process** (*C1-CC-INITOP*) that can be plugged in on the Collection Class Overdue Rules. For more information on how account monitor rules can be configured for overpayment, see *Overdue Rules Are Embodied In Algorithms*.
- An overpayment process can be created manually by selecting **Main Menu** > **Accounting** > **Overpayment Process**.

An Overpayment is Typically Reviewed

When an overpayment is identified, it typically goes through a review process, which may include a combination of automated checks and user review and/or approval.

Examples of review activities include:
- Processing Rules - these can include checking for a valid mailing address, verifying bank account details.
- Compliance Rules - these include specific criteria defined by the tax authority for overpayments that require additional review.
- Minimum Balance - the overpayment amount is compared against a minimum threshold amount to determine whether the overpayment should be processed.
- Risk-based thresholds and overrides - if the overpayment amount is greater than a predefined maximum threshold, it may require special handling and additional approval.
- Suppression - an overpayment may be suspended if an account is under investigation for other debt or another reason.
- Separation of Duty - the tax officer approving the refund must be different from the tax officer who changed the account's bank information.

The review rules may differ with each tax authority and tax type. However, the end result is the same: to determine whether the overpayment can be processed further.

Not all Overpayments Result in a Refund

Before an overpayment is refunded to the taxpayer, some or all of the overpaid amount may be offset against other existing debt, carried forward to a future period, and/or contributed to designated charitable funds or organizations.

Calculating Interest

Before any other action is taken, it might be required to calculate interest on the overpayment amount. The calculation rules will differ by tax authority and by tax types. The base-package is supplied with the following algorithms to calculate interest:
- Calculate interest using a rate-able adjustment (*C1-OP-CALINT*)

- Calculate interest using a rate factor (*C1-OP-INTRF*)

Offset

When an overpayment is offset, the credit is used to extinguish other debts. The order of allocation of overpayment amount to debt depends on the tax authority's rules. Typically, the credit is applied in the following order:

- Within the same assessment
- Other assessments within the obligation
- Other obligations under the same account
- Other accounts for the taxpayer

In some cases, a tax authority may also allow offsetting against debt from other agencies. This type of debt is called external debt.

Where offset is possible, it is done before attempting to contribute, carry forward or refund. Offset is not allowed on future obligations, but a carry forward is done instead.

The base-package is supplied with an algorithm (*C1-OP-OFFST*) to offset credit to other obligations.

Carry Forward

Taxpayers may opt to pay an amount of an overpayment to a future period. This is known as carry forward. This is common to individual income tax filing, where the taxpayer can indicate on the form a designated amount that will go to the next filing period (next tax year).

A tax authority's rules may designate certain tax types to only allow overpayments to be carried forward. Some common examples include excise or sales and use taxes. In these cases, the entire overpayment amount is carried forward.

The base business object includes an algorithm (*C1-OP-CAFUFP*) to carry forward to a future period.

Contributions

Most tax forms provide an option to contribute to a charitable fund/organization. The taxpayer indicates a specific amount to contribute to each selected fund/organization. The contribution is made only if an overpayment exists.

The base business object includes an algorithm (*C1-OP-ACNAMT*) to create contributions.

Refunds

Refunds result after attempts to offset, contribute and/or carry forward. The credit amount is given back to the taxpayer in the form of a paper check or a direct deposit. The taxpayer could either authorize a one-time direct deposit by putting bank account information on the tax return or indicate a recurring direct deposit for all refunds. In the case of a recurring direct deposit, the bank account information stored for the taxpayer on the *account* will be used.

The base business object includes an algorithm (*C1-OP-CREREF*) to create a refund.

Individual Taxpayer Overpayment Process

The base product includes the business object **C1-OvrpyProcAutoCredRef**, which is designed to cater for overpayment processes typical of individual taxpayers.

The Overpayment Process Lifecycle

The lifecycle of the overpayment process depends upon the configuration of the associated business object. The Individual Taxpayer Overpayment Process has a lifecycle typical of overpayments for individual taxpayers:

- The overpayment is initially validated against a number of business rules. The overpayment amount would typically be compared against a minimum threshold write-off amount.

- If issues are detected during validation, the overpayment will transition to an issues detected/error state. The user will be able to see a list of issues and correct them.
- The overpayment may require approval(s). The threshold amounts and approval roles will be defined on the overpayment type. A user can approve or reject the overpayment.
- Once all the required approvals have been obtained, certain actions will take place such as offset, carry forward, contributions and/or refund.

Note:

Refer to the **C1-OvrpyProcAutoCredRef** Business Object metadata for more details.

Overpayment Process Approval

An overpayment process typically requires one or more levels of approval before any financial activity can take place.

An Overpayment Process Typically Requires Approval

If the overpayment process type defines an approval hierarchy, users will need to approve the overpayment process in order for the process to continue. When the overpayment process is in the **Approval in Progress** state, two action buttons will appear: **Approve** and **Reject**.

If the user chooses to reject the overpayment process, they will be prompted for a reject reason.

To Do Entries Are Created To Notify Approvers

When the overpayment process detects an approval is required, 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 overpayment process type. All users who belong to the approving To Do role can see the entry. When a user drills down on an overpayment approval To Do entry, the *overpayment process* portal is opened. This portal contains summary information about the overpayment process. This portal is also where the user approves or rejects the overpayment process.

When the user approves the overpayment process, the To Do entry is **Completed** and the overpayment process's log is updated. If there are no higher levels of approval required, the overpayment process will transition 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.

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 overpayment processes requiring their attention. Refer to *To Do Entries May Be Routed Out Of The System* for the details.

Monitoring and Escalating Overpayment Approvals

The base-package is supplied with an algorithm that your implementation can use to monitor overpayment 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-OP-CHKOTO* 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-OvrpyProcAutoCredRef** business object.

Rejecting Transitions the Overpayment Process to a Final State

When an overpayment process is being approved, anyone with the right level of access can reject it by using the *overpayment approval* zone.

When an overpayment process is rejected, the following takes place:

- The user is prompted for a reject reason.

- The overpayment process's log is updated with the reject reason and the overpayment process is moved to the **Rejected** state.

Enabling the System to Use Individual Income Overpayment Process

To enable this functionality the following configuration tasks are needed:

- Various adjustments are used for calculating interest, write off and offset for the overpayment process. For each of these actions, a suitable *adjustment type* is required.
  - If you wish to specify a minimum amount threshold for the overpayment process, you will need to define a non-calculated adjustment type for the minimum amount write-off.
  - If your implementation rules specify that interest should be calculated and the interest is calculated using a *rate*, you will need to define the following:
    - A calculated adjustment type for offset-able interest, or a calculated adjustment type for non offset-able interest.
    - An associated rate schedule that includes a *RQ rule* for calculating number of days
  - If your implementation rules specify that interest should be calculated and the interest is calculated using a *rate factor*, you will need to define the following:
    - A non-calculated adjustment type for offset-able interest, or a non-calculated adjustment type for non offset-able interest.
    - An associated rate factor.
  - If your implementation allows offset, then a non-calculated adjustment type needs to be defined for the offset.
  - To allow refunds via a check, an *A/P adjustment type* needs to be defined.
- Make sure the adjustment types are defined on an associated *adjustment profile* that is linked to any obligation types that are going to be covered by the overpayment process.
- To allow refunds via direct debit a *distribution rule* needs to be defined. The distribution rule should be designed to pay a specific obligation.
- If approvals are required as part of the overpayment process, a To Do type is needed. The base product is supplied with a To Do type called *C1-OVAPP* that should be used as the basis for approval To Do type.
- If you wish to have a To Do created when the overpayment process is automatically cancelled, you will need a To Do type. Note that the base-package is supplied with a To Do type called *C1-OVPYX* that should be used as the basis for automatic cancellation To Dos.
- If you wish to have a To Do created when issues are detected during the validation of the overpayment process, you will need a To Do type. Note that the base-package is supplied with a To Do type called *C1-OVISS* that should be used as a basis.
- For each To Do type that you wish to use, you will need a To Do role.
- The system has been currently configured to calculate non offset-able interest using a rate factor. If you wish to change this you will need to inactivate the base-package algorithm by adding an appropriate inactivation option to the business object. You will then need to plug in a different algorithm based on the logic you wish to implement.
- Define an overpayment process type for the business object **C1-OvrpyProcTypeAutoCredRef**.

Implementing Other Overpayment Process Types

The above sections describe how the base-package overpayment process for individual taxpayer works using the **C1-OvrpyProcAutoCredRef** and **C1-OvrpyProcTypeAutoCredRef**business objects. Your implementation can add additional business rules and change the overpayment process user interface as required. Alternatively, if your implementation needs a different overpayment process, you can create different business objects with their own business rules.

Setting Up Overpayment Options

Setting Up Overpayment Process Types

An Overpayment Process Type defines the configuration information that is common to overpayment processes of a given type. The type of information captured on the overpayment process type is governed by the overpayment process type's business object.

To set up an Overpayment Process Type, select **Admin Menu** > **Overpayment Process Type** .

The topics in this section describe the base-package zones that appear on the Overpayment Process Type portal.

Overpayment Process Type List

The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent overpayment process type.
- Click the **Add** link in the zone's title bar to add a new overpayment process type.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Overpayment Process Type

The Overpayment Process Type zone contains display-only information about an Overpayment Process Type. This zone appears when an Overpayment Process Type has been broadcast from the Overpayment Process Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

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

Overpayment Process Type Log

This is a standard *log zone*.

Defining Forms Rules

Forms processing is a core function for a tax authority. When forms are processed, the system needs to verify that the information is correct and update or create the appropriate taxpayer details and transactions.

Form rules provide business users with the ability to configure processing rules for a form type.

Understanding Form Rule Administration

This section describes concepts and common tasks related form rule administration.

About Form Rules

A form rule is combination of configuration data and processing logic that applies a business rule. The configuration that controls the form rule logic is defined on the form rule business object. The form rule BO has an associated **Apply Rule** algorithm to execute the logic of the rule.

When forms are defined, each form section and line must reference a business name that is unique for that form type. As form changes are introduced, these business names remain constant.

Many form types within a given set of forms also have common information such as taxpayer identification, income and credit lines and so on. The validation and processing rules are typically the same for these common components.

Form rules may be designed to reference section and line business names rather than specific form lines. If a section / line exists in the form from one year to the next, its rules can be reused. Form lines that are common across form types can utilize the same rules provided the business names are the same.

Fastpath:

Refer to *Designing Your Form Rules and Groups* for information about designing your rules.

About Form Rule Groups

A form rule group is a uniquely named group of rules. You can predefine a set of rules common to a particular form section or set of form lines that can be selected and associated with a form type. For example, predefined rule groups can be used to verify that all taxpayer identification details have been filled in on a form or complete the actions required when a form is posted.

Form rule groups have an associated reference form type. Reference form types are used to determine the section and line business names that may be used by rules in this group to reference the form data. The sections and line names defined in the reference form type must be common to all form types that use this rule group.

Form rule groups are also used to define the form rules for a form type. A form type can define one or more form rule groups to be executed during certain steps in the form lifecycle (defined by a rule event). Sequence numbers are used so that multiple form rule groups executed for the same rule event are done in an appropriate order. Note that the system looks to the specific form type for the form being processed to determine which rules to execute, not the reference form type on the rule group. This means that the rules to be executed can be tailored to the different form types within a broader category of forms.

Fastpath:

Refer to *The Big Picture of Rule Processing* for more information about processing different rule events.

About Exception Categories

An exception category is a broad categorization of exceptions that can be reported by form validation rules. An exception category must be specified on a form exception.

The Big Picture of Rule Processing

As a form transitions through the states in its lifecycle, various rules need to be executed. The following sections describe the way in which the base system processes rules.

Apply Rules Overview

The following section highlights the logic in the base algorithm type *C1-FRM-APPRL*. It has been designed to call the algorithms plugged into the Apply Rule plug-in spot for each rule linked to the form type for a given rule event. This

algorithm type should be plugged into the **Business Object Status - Enter** plug-in spot for each state that requires rules to be executed.

At a high level, the base algorithm follows these steps

- Read the form data and store it in memory
- Find the form rule groups linked to the input form type and rule event. For each group in sequence order, retrieve its **Active** rules in sequence order.
- For each rule,
  - Execute its Apply Rule algorithm
  - Check the returned value of Rule Action. If the action is **Terminate**, terminate overall processing; otherwise continue on to the next rule.
- When processing is complete,
  - Update the form using the data updated in memory by the rule algorithms
  - If exception processing applies and the exception list is populated,
    - Compare the current list of exceptions to the existing exceptions for the form, if any. Any exceptions that are no longer in the current list are closed. Add any newly-reported exceptions to the form exception table.

Note:

No updates. The rules processing algorithms provided in the base product do not do any updates to the database. These algorithms simply update the internal form and exception information with new values. The rules processing algorithm is responsible for updating the form and creating or closing form exceptions.

The Big Picture of Validation Rules

Verifying Form Data Using Form Rules

Validation rules determine if the information supplied on the form can be processed. This usually includes checking that required information is supplied and validating that the data is in the correct formats. In some cases, it may mean correcting the form data within certain tolerance limits. Validation rules are responsible for reporting any issues on a form that may require correction.

Most forms have a lifecycle that includes a validation step. The base package provides a sample tax form **C1-ParentTaxForm** that has a **Validate** status. This status has an enter-plug in that uses the base *C1-FRM-APPRL* algorithm type to execute rules for the **Validation** rule event and process any exceptions reported.

Reporting Exceptions

Algorithms that perform validation rules are responsible for reporting any issues as a form exception. Form exceptions details must include the form rule group and rule that detected the issue, an exception category and an exception class. The base package includes exception class values of **Notification**, **Waiting for Information** and **Suspense**. Your implementation can add further classes as appropriate for your business rules.

Stopping Form Processing When There Are Issues

A form can suspend if there are errors that a tax authority user needs to review or fix.

If the form is missing key information, the form can go into a 'waiting' state until the tax authority receives the information from the taxpayer.

The exception class attribute can be used to determine what state a form should move to if there are open issues after validation. The base package algorithm type *C1-FRE-TROEX* is designed to transition a form to a waiting state if open exceptions of class **Waiting for Information** exist or to a suspended state if there are open exceptions of the

class **Suspense**. This algorithm type should be plugged into the **Business Object Status - Enter** plug-in spot after the**Apply Form Rules** algorithm.

The Big Picture of Processing Rules

Processing Form Data Using Form Rules

Once a form's data has been verified, it may be posted into the system. A posted form may subsequently be adjusted to correct information, transferred to another obligation or account or reversed altogether. At each of these steps in the lifecycle, other actions need to take place. The system may need to create taxpayers, accounts and obligations, create financial details, create overpayment processes or possibly reverse the effect of previous updates.

The base package algorithm type *C1-FRM-APPRL* is designed to be used on any state that requires rules processing. To use this algorithm type to execute processing rules, you need to define a rule event applicable to that state and plug in an algorithm configured to execute rules for that event. The base package includes the following pre-configured rule events and associated algorithms; **Suspend, Waiting for Info, Post, Adjust, Transfer** and **Reverse**. The base package provides a sample tax form **C1-ParentTaxForm** that may be used as an example of a form that implements rule processing using these rule events.

Processing Rules Create Other Objects

Algorithms that perform processing rules are responsible for any data updates that affect other objects in the system such as accounts, obligations, adjustments and others. As such, they are also responsible for creating form log entries that link those objects to the form being processed. Refer to the base algorithm *C1-FR-CRASMT* for an example of a rule algorithm that creates an assessment adjustment and its associated log entry.

Processing Rules Do Not Report Exceptions

In a typical form lifecycle, the actions that take place when a form is posted, adjusted, transferred or reversed are interrelated. If one of these actions is not successful, an error should be reported and the form should remain in the same state as before the updates took place. For this reason, the system does not support exception handling after processing rules are executed. Any issues that are encountered should be reported using standard error handling.

Setting Up Exception Categories

When a form's data is validated the system may identify errors which it stores as exceptions. You can use exception categories to group these exceptions. The exceptions for the forms can then be viewed on the Exceptions tab of the form's maintenance portal. To maintain an existing Exception Category, select **Admin Menu** > **Exception Category**.

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

Exception Category List

The Exception Category *List zone* lists every Exception Category. The following functions are available:

Click the *broadcast* icon to open other zones that contain more information about the adjacent Exception Category.

The standard actions of **Edit** and **Delete**are available for each Exception Category.

Click the **Add** link in the zone's title bar to add a new Exception Category.

Actions

This is a standard actions zone. The **Edit** and **Delete** actions are available.

Exception Category

The Exception Category zone contains display-only information about an Exception Category. This zone appears when an Exception Category has been broadcast from the Exception Category List zone or if this portal is opened via a drill down from another page.

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

Setting Up Form Rule Groups

Form rules are logically grouped together in a form rule group. The rules are executed in the following order: Each rule group linked to the form type is executed in sequential order. The rules in each group are executed in sequential order.

To set up a form rule group, open **Admin Menu** > **Form Rule Group**.

Form Rule Group Query

Use the query portal to search for an existing form rule group. Once a form rule group is selected, you are brought to the maintenance portal to view and maintain the selected record.

You can also click the Add link in the query zone's title bar to add a new form rule group.

Form Rule Group Portal

This portal appears when a form rule has been selected from the Form Rule Group Search results.

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

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Form Rule Group

The Form Rule Group zone contains display-only information about a form rule group. This zone appears when a form rule group has been selected from the Form Rule Group Search results, or if this portal is opened via a drill down from another page.

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

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

Form Rules List

The Form Rule *List zone* provides a summarized view of all form rules for a form rule group. A row is displayed for every form rule linked to the form rule group. The rules are shown in sequential order. The list includes the form rule information and a hyperlink to the rule's maintenance portal.

The standard actions of **Edit, Delete** and **Duplicate** are available for each form rule.

Click the **Add** link in the zone's title bar to add a new form rule.

Form Types

This zone displays all form types that reference the form rule group. This zone only appears if a form rule group exists in the portal context. The list includes the name of the form type and a link to the form type's maintenance portal.

Setting Up Form Rules

A form rule is designed to implement one specific rule used in processing a form. Examples of form rules include taxpayer existence rules, form line validation, calculation and auto-correction rules, tolerance checking, and posting logic.

To set up a form rule, open **Admin Menu** > **Form Rule**.

Form Rule Query

Use the query portal to search for an existing form rule. Once a form rule is selected, you are brought to the maintenance portal to view and maintain the selected record.

You can also click the Add link in the query zone's title bar to add a new form rule.

Form Rule Portal

This portal appears when a form rule has been selected from the Form Rule Search results.

Note:

Note: When the Form Rule portal is used the context-sensitive Rules on Group zone appears in the dashboard. This zone lists the other form rules in the same group as the rule in context. From this dashboard zone you can add a new form rule to the group, or go to the maintenance portal for the other rules in the group

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

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Form Rule

The Form Rule zone contains display-only information about a form rule. This zone appears when a form rule has been selected from the Form Rule Search results, or if this portal is opened via a drill down from another page.

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

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

Designing Your Form Rules and Groups

The base product provides a generic BO for Form Rule Group, **C1-FormRuleGroup**. Your implementation can add additional business rules to this BO as required.

The base product provides a parent BO for Form Rule, **C1-FormRule**. This BO defines the common elements for all form rules. It is not intended to be used stand-alone. Your implementation can include this BO in each rule BO and add the additional components such as custom UI Maps as required.

The base product supplies a number of BOs that apply particular Form Rules. Refer to the business objects that reference the Form Rule maintenance object **C1-FORMRULE** for a complete list.

Your implementation can define additional form rule BOs as required. The following points highlight the important configuration for this business object:

- Develop the appropriate rule processing algorithm to perform the rule logic.
- For any configuration required by the rule processing algorithm, consider whether it should be defined when configuring the form rule by a business user. If so, include the appropriate elements in the form rule business object's schema.
- For a rule that performs validation, consider whether the rule exception information, which includes the Exception Category should be defined when configuring the form rule by a business user. If so, include the appropriate elements in the form rule business object's schema.
- Consider what logical grouping this rule belongs to. Note that a group of rules is always executed in the context of a single rule event. Rules should be grouped according to common actions that make take place for a particular event or set of events.

Using The Conditional Element Rule

The base product provides a Form Rule BO that is designed handle a large range of conditional validation, **C1-CondElementValidation**.

The **Conditional Element Validation** BO allows a business user to define a series of conditions that apply to sections and lines of a form. The conditions are defined as basic mathematical expressions that have a result of true or false. Your implementation should consider using this rule BO when defining a rule that validates a form line value in relation to other supporting lines on the form.

The following topics highlight some of the available features for conditional expressions. Other mathematical operators and functions are supported but these are not normally used for calculations that are common to forms validation.

Variables

Variable identifiers must be a character between a-z except x. Variable x is always used to refer to the form line that this rule applies to. Variable identifiers are case sensitive.

Operators

The following types of expression operators are supported.

Mathematical Operators

Operators of *, /, %, +, - are allowed. Examples of expression using these operators are:

- **x** = **a** + **b** + **c** where the value of **x** should equal the sum of the values **a**, **b** and **c**
- **x** = **a** * **r** where **r** is a rate factor variable

Comparison Operators

Operators of =, != (not equal), <, >, <= and >= are allowed. Examples of expression using these operators are:

- **x** > **a** where the value of **x** should be greater than the value of **a**
- **x** != 0 where **x** must have a non-zero value

Grouping Operators

Parentheses may be used to formulate expressions with sub-computations. An example of an expression using grouping is:

- **x** = (**a** + **b**) / **c** where the value of **x** should be equal to the result of adding values **a** and **b** then dividing by the value of **c**

Functions

The following examples show the use of the most common supported functions:

Ceiling	**x** = ceiling (**a*** **b**) where **x** should equal the result of multiplying the value of **a** by the value of **b** and rounding up to the nearest whole number
Floor	**x** = floor (**a*** **b**) where **x** should equal the result of multiplying the value of **a** by the value of **b** and rounding down to the nearest whole number
Round	**x** = round (**a*** **b**) where **x** should equal the result of multiplying the value of **a** by the value of **b** and rounding the nearest whole number. A result of 1.5 would round to 2 and a result of 1.4 would round to 1.
Any	any [value in **a** | value !=''] where **a** is a value from a repeating line and the condition is true if any of the values of **a** are not blank
All	all [value in **a** | value !=''] where **a** is a value from a repeating line and the condition is true if none of the values of **a** are blank

Defining Obligation Types

Every obligation must reference an obligation type. The obligation type defines the responsibilities between the taxpayer and the tax authority. The obligation type dictates how returns, bills and payments are handled as well as how overdue debt is collected and how tax collections will be booked in your general ledger.

Contents

Designing Obligation Types

The topics in this section provide guidelines describing how to design obligation types. When designing obligation types you will want to consider what the obligation will be used for. Generally speaking, obligation types can be separated into two categories.

- The first type of obligation type relates to the specific tax types of a tax authority. Examples of these include corporate tax and property tax. The specific tax type dictates how bills are handled and what collections activities are available to the tax authority.
- The second type of obligation relates to specific events or transactions that require certain follow-up actions from the tax authority or taxpayer (or both). Examples of these include overpayment obligations and payment plan obligation. With overpayments, the tax authority has the responsibility to properly handle the excess credit a taxpayer has created with a payment. Payment plans are the responsibility of the taxpayer in that they are required to send in a series of payments on a predefined schedule. Each obligation must be monitored to ensure proper compliance.

Filing Period Obligations

Filing period obligations are one of the tax type specific obligation types. Examples of these tax types include individual income, sales and use, corporate, withholding, and fuel taxes. These are examples of return based taxes in which the taxpayer has the responsibility to file a return for each filing period obligation. There are usually statutory considerations when configuring these obligation types. You will want to consider the following when designing your obligation for these types of taxes.

- The obligation type must reference a defined *tax type*. The specific tax type will dictate the rules for that obligation. It should indicate that a Tax Role is required and should include the valid filing calendars for the tax type.
- The obligation type should be configured to indicate that a filing period is required.
- Distribution code and general ledger division for the obligation directs how the money is accounted. Different taxes may be moved to different accounts. While an individual income tax may go to the general treasury fund, a fuel tax may go the highway construction fund. The distribution code will reference an algorithm that directs how the money will be accounted for.
- Adjustment profile must be defined for the obligation. Adjustment profiles contain the individual listing of adjustment types that are available for that obligation type.
- An appropriate P&I control and P&I algorithms should be defined for the obligation type. Refer to *P&I Rules for a P&I Control Define the Calculation* for more information.
- Define the default debt category priority for credits linked to obligations of this type. Refer to *Debt Categories and their Priorities* for more information.

Each tax type will have its own unique attributes that will change how the obligation is set up. The more details you add to your obligation type, the more robust your processing of that obligation will be. Before configuring for this obligation type you should thoroughly review the statutory considerations and make sure they are all addressed when setting up your obligation type.

Property Tax Obligations

Property tax obligations are another type of tax specific obligation type. Property tax obligations are based off the assessed value of a taxpayer's asset. Typically they are items such as real estate or automobiles but can include boats, RV's, and other high value items. They differ from filing period obligations in that it is the tax authority's responsibility to assess the value and calculate the amount of tax owed. It is then the taxpayer's responsibility to either pay or appeal the assessed value. The considerations are very similar to filing period obligations.

- The obligation type must reference a defined tax type. The specific tax type will dictate the rules for that obligation.
- Payment plans are often utilized for property tax obligations. Due to their infrequent assessment and billing (once or twice a year) the amount is usually large enough where a significant number of taxpayers would have difficulty paying. If payment plans are utilized then make sure to check that option under the billing tab.
- Distribution code and general ledger division for the obligation directs how the money is accounted. Different taxes may be moved to different accounts. While an individual income tax may go to the general treasury fund, a fuel tax may go the highway construction fund. The distribution code will reference an algorithm that directs how the money will be accounted for.
- Rates may be utilized more often for property taxes. Property taxes tend to be a locally administered tax. The breakdown of how the tax is applied to local services can be done using the rate schedule option under the rate tab.
- Adjustment profile must be defined for the obligation. Adjustment profiles contain the individual listing of adjustment types that are available for that obligation type.

Billable Charge Obligations

You create a billable charge whenever a taxpayer should be charged for a service that occurs outside the normal course of business. For example, if a taxpayer requires a review of their property assessment, you may charge them an administration fee. 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 an obligation. This obligation behaves just like any other obligation:

- **Bill segments are created for the obligation.** Whenever billing is performed for an account with billable charge obligations, the system creates a bill segment for each obligation with unbilled charges. If multiple unbilled charges exist for a given obligation, only one bill segment will be created and it will contain details about all of the billable charges.
- **Payments are distributed to the obligation.** Payments made by an account are distributed to its billable charge obligations just like any other obligation.
- **Overdue debt is monitored.** The credit and collections process monitors billable charge obligations for overdue debt and responds accordingly when overdue debt is detected.

Therefore, you must set up at least one obligation 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 obligation 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 obligation type. If not, you'll need one obligation type for each combination.

Overpayment Obligations

When a taxpayer 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 obligation to hold the excess (let's call it an overpayment obligation). The credit would need to be transferred from this obligation to the tax obligations at an appropriate time.
- You could direct the excess payment on one of the existing tax obligations.

You control which method is used by plugging in the appropriate **Overpayment Distribution** algorithm on each *account type* (i.e., you can choose a different method for different account types). If you choose to hold overpayments on a separate obligation, then you must set up an obligation type.

The following points highlight interesting information about overpayment obligation types:

- It should have an accounts payable distribution code. This is because when a payment segment is created for this type of obligations, the system must credit a liability (an overpayment is a liability).
- It's important to indicate that the overpayment obligation is a one-time obligation. Why? Because this means that the system will automatically close the obligation when it's balance falls to zero (i.e., when appropriate business processes transfer all of the overpayment to tax obligations).
- A bill segment type is not needed because the system never creates bill segments for such obligations (they exist only to hold excess credits).
- P&I controls and the P&I related algorithms are not needed.
- You may also want to turn on the alert message
- You must reference this overpayment obligation type as the parameter value on your overpayment algorithm (this algorithm is plugged in on the desired account types). Refer to *Overpayment Algorithm* for more information about this algorithm.
- You must design appropriate business procedures to use this overpayment when additional debt is incurred on tax obligations.

Write Off Obligations

Some agencies may choose to transfer written of debt from the "normal" obligation onto one or more write-off obligations. When the debt is transferred to a write-off obligation, the distribution code on the "normal" obligation is credited (typically an A/R GL account), and the distribution code on the write-off obligation is debited.

You will almost always need a write-off obligation whose distribution code is the write-off expense. However, if you don't practice cash accounting, you may have uncollectible debt for liabilities (and you don't owe the liability if you don't get paid). In this case you'll need another obligation type for the liabilities.

The following points highlight interesting information about write-off obligation types:

- The distribution code is either an expense or a liability account. This is because when debt is transferred to these types of obligations, 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.
- They do not need a bill segment type because the system never creates bill segments for such obligations (they exist only to hold uncollectable debt)

Note:

The adjustment type used to set the offending obligation's current balance equal to its payoff balance is defined on each write-offable obligation type. The adjustment type used to transfer the delinquent debt to the write-off obligation is defined on the write-off obligation type.

Note:

An Alternative. If you have a limited number of liability accounts, you may decide to have a separate write-off obligation for each liability account. Doing this would proliferate the number of obligations created at write-off time. However, it would simplify the remittance of payment to the third party if the reversed liability is ever paid.

Over/Under Cash Drawer Obligations

In order to balance a tender control that is out-of-balance, your organization must set up an account with an obligation whose obligation type references the over/under expense account. You will probably only have one obligation that references this obligation 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)*.

The following points highlight interesting information about this obligation type:

- It has an expense distribution code. This is because when a payment segment is applied to this type of obligation, 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 obligations (it only has over/under payment segments linked to it).

Payment Upload Error Obligations

If the payment upload process detects an invalid account on a payment upload record, it will create a payment for the suspense obligation defined on the upload process' tender source (see *Setting Up Tender Sources*). You should create a special obligation type for this obligation.

Fastpath:

For more information about the payment upload process, refer to *Phase 3 - Create Payment Events, Tenders, Payments and Payment Segments*.

The following points highlight interesting information about this obligation type:

- It has an expense distribution code. This code should probably be a suspense account. All payment segments that are created for this obligation will eventually be transferred to a "real" obligation 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 obligations (it only has invalid account payment segments linked to it).

Pay Plan Obligations

If you allow your customers to pay overdue debt using a payment plan, you need to set up obligation types for pay plan obligations.

Fastpath:

For more information about pay plans, refer to *Defining Pay Plan Options*.

Defining Obligation Information

The obligation information displayed throughout the system is controlled by a plug-in.

The system first looks to see if the obligation type references an obligation business object and if the business object defines an information plug-in. If a BO is not provided or if that BO does not define an information algorithm, the system looks for an information algorithm plugged into the obligation maintenance object.

If no plug-ins are found on the BO or the MO, the system looks for a plug-in algorithm on the *Obligation Type*.

If such an algorithm is not plugged-in on the Obligation Type, the system looks for a corresponding algorithm on the *installation record*.

If you prefer different formatting logic, your implementation should provide a plug-in at one of the above plug-in spots, as appropriate for your business rules.

Setting Up Obligation Types

In this section, we explain how to create/maintain your obligation types.

Contents

Obligation Type - Main Information

Open **Admin Menu, Obligation Type** and navigate to the **Main** tab to define core information about your obligation types.

Description of Page

Enter a unique combination of **Division** and **O bligation Type** for every obligation type.

Enter a **Description** for the obligation type.

Tax Type defines the type of service associated with the obligation type. If the obligation type has rates, only rates belonging to this tax type may be linked to the obligation type.

Fastpath:

For more information about tax types, refer to *Setting Up Tax Types*.

Use an **Obligation Business Object** to define a *BO* that may govern additional rules related to obligations of this type.

Indicate if the **Filing Period Applicability** for obligations of this type is **Required** or **Not Allowed**.

Select the **Distribution Code** and **GL division** that defines the receivable account for receivable-oriented obligations. For non-receivable oriented obligations, this distribution code depends on the obligation type's function. Refer to *Designing Obligation Types* for some examples.

Select the **Revenue Class** associated with the obligation type (and its obligations). The revenue class may affect the revenue account(s) generated by the obligation's rate.

Fastpath:

Refer to *Rate Component - GL Distribution* for more information about revenue class.

Select the **Payment Segment Type** that defines how payment segments linked to obligations of this type affect:

- The obligation'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 *account type*. Because adjustments must be linked to an obligation, the algorithm must determine the appropriate obligation to use to levy the adjustment based on business rules. The charge is levied using the **NSF Adjustment Type** of the appropriate obligation's obligation type.

Caution:

You must specify adjustment type profiles on the obligation 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*.

Define the **Payment Grace Days** applicable for obligations of this type. Refer to *Due Dates for Filing Period Based Obligations* for more information about due dates.

Select the **Payment Priority**. This field is available for use by the algorithms that distribute partial payments amongst an account's obligations. Higher priority obligations 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:

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 obligations*.

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 obligations. Higher priority obligations 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:

Note. The values for this field are customizable using the Lookup table. This field name is DEL_PRIORITY_FLG.

Define the default **Debt Category Priority** for credit financial transactions linked to obligations of this type. Refer to *Debt Categories and their Priorities* for more information.

Turn on **Do Not Overpay** if the system is not allowed to distribute an overpayment to this type of obligation (i.e., the obligation 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 Obligation* for information about an overpayment algorithm that uses this field.

Where Used

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

Obligation Type - Detail

Open **Admin Menu, Obligation Type** and navigate to the **Detail** tab to define additional details about a given obligation type.

Description of Page

Turn on **Display As Alert** if Control Central should display an alert if an account has an obligation 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 obligation types (e.g., write-offs) so that a CSR is not presented with an alert for every obligation type.

If this obligation type is used for any of the special roles defined in the drop down of **Special Role Flag**, indicate which one. Valid values are: **Billable Charge**, **Pay Plan, Write Off**. This information is used on windows with functionality that can only be used by obligations used for specific roles. For example, the Billable Charge window group can only reference **Billable Charge** obligations.

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 obligations to the write off obligation.

Caution:

You must specify adjustment type profiles on the obligation type (on the Adjustment Type window) before adjustment types will appear in the above drop downs.

Turn on **One Time Charge** if this obligation type is used for one-time invoices. When a one-time invoice obligation is created, the system sets the stop date of the obligation to be equal to the start date.

Where Used

The alert information is used by Control Central to alert a CSR when unusual obligations exist for an account. Refer to *Control Central - Main* for more information.

Only obligation types designated as being **Billable Charge** may have billable charges linked to them. Refer to *Maintaining Billable Charges* for more information.

Only obligation types designated as **Pay Plan** may be used to set up pay plans. Refer to *Pay Plans* for more information.

Only obligation types designated as being **Write Off** may be specified as the write off obligation type on distribution codes. Refer to *Setting Up Distribution Codes* for more information.

Obligation Type - Billing

Open **Admin Menu, Obligation Type** and navigate to the **Billing** tab to define how the system manages bill segments for obligations of a given obligation type.

Description of Page

Turn on **Eligible for Billing** if the system should create bill segments for obligations of this type.

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 obligation type will be created and how the related financial transaction affects the general ledger and the taxpayer's debt.

Fastpath:

For more information about bill segment types, refer to *Setting Up Bill Segment Types*.

Define the default **Debt Category** to assign to bill segment type financial transactions for obligations of this type. Refer to *Debt Categories and their Priorities* for more information.

Note:

Required for base algorithms. The base P&I calculation algorithm and the base *Determine Detailed Balance* algorithm require that every debit financial transaction reference a debt category.

Use **Default Description on Bill** to define the verbiage that should print on the taxpayer's bill.

Note:

Rates overwrite this description. The Default Description on Bill is not applicable for obligations 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 obligations 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 obligations linked to this obligation type are processed by billing processes.

Use **Bill Print Priority** to define the order in which the obligation type's bill segments should appear on bills (relative to the other obligation types that appear on a bill).

Note:

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 obligations of this type when they are first created. An operator may change the default value on an obligation in case a specific taxpayer 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 obligation type and there already exist obligations of this type, the existing obligations will contain the original value (the new value on the obligation type will not be propagated on the existing obligations).

Turn on **Characteristic Location Required** if a characteristic location must be linked to the obligation when the obligation is activated. The characteristic location is used to define the taxing authorities associated with the obligation's bill segments. It is also used to identify where the obligation's service is located on various windows.

Fastpath:

For more information about how characteristic location is used, refer *An Illustration Of A Rate Factor And Its Characteristics*.

Use the **Initial Start Date Option** to control how billing should calculate the calculation period for the very first bill for obligations of this type. This field is important if your obligation has rate components that include daily charges. Set the field to **Include First Day** if the obligation's start date should be included in the daily charges. Set the value to **Add 1 Day Always** if the obligation's start date should never be included in the daily charges. Set the value to **Add 1 Day for Back-to-back** if the obligation's start date should only be included in the daily charges if no previous taxpayer was responsible for the charge (for example for property tax). This field is not applicable for obligation types with a special role of **Billable Charge**.

Obligations 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 obligation 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 *Defining Bill Cycles and 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, obligations may have their bill segment end date based on a specified date. If this obligation type should be billed like this, select **Anniversary Future Billing** or **Anniversary Past Billing** in the **Use Calendar Billing** field. When either option is used, you must define the **Anniversary Bill Frequency**. This frequency defines the amount of time between bill segments.

Fastpath:

For more information about anniversary billing, refer to *Using The Anniversary Method*. For more information about other bill end date methods, refer to *Ways To Control The End Date Of A Bill Segment*.

Total Bill Amount indicates whether obligations of this type can use the total amount to bill field on the obligation page. Valid values are **Not Allowed** and **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 obligation page for obligations of this obligation type.

Recurring Charge indicates whether obligations of this type can use the recurring charge field on the obligation window. Valid values are **Not Allowed**, **Optional** and **Required**. If either **Optional** or **Required** are used, you must enter:

- **Recurring Char ge Amoun t Label**. This defines the label that prefixes the recurring charge amount on the obligation window for obligations of this obligation type.
- **Recurring Charge Frequency**. This defines the following:
- Specifies the frequency at which the Recurring Charge Amount specified on obligations of the obligation type is to be billed.
- Serves as the basis for proration of the Recurring Charge Amount.
- Specifies the frequency at which obligations of the obligation type without a rate will be billed.

Set the **Eligible for Pay Plan** flag to **Eligible** if you want obligations of this type to be eligible to be covered by a pay plan.

Obligation Type - Rate

Open **Admin Menu, Obligation Type** and navigate to the **Rate** tab to define the rates that may be referenced on obligations of a given type.

Description of Page

Turn on **Rate Required** if the bill segment creation algorithm for the obligation type expects a rate schedule to be referenced on obligations of this type.

Fastpath:

For more information, refer to *Rates*.

Define the date the system uses when selecting an effective-dated rate (from the obligation'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 calculation 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 obligations of this type. The following fields are required for each obligation type:

Rate Schedule	Specify the rate schedule; its description is displayed adjacent.
Use Rate As Default	Turn on this switch for the rate to be defaulted on new obligations.

Where Used

This information is used to default and validate the rate specified on an obligation. Refer to *Obligation - Rate Info* for more information.

Obligation Type - Adjustment Profiles

Open **Admin Menu, Obligation Type** and navigate to the **Adj Profile** tab to define the adjustment profiles that define adjustment types that may be referenced on obligations of a given type.

Description of Page

Define the **Adjustment Type Profiles** that, in turn, define adjustment types that may be referenced on obligations 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 obligation. Refer to *Adjustments - Main Information* for more information.

Obligation Type - Characteristics

To define characteristics common to all obligations of a given type, open **Admin Menu, Obligation 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 obligations of this type.

Note:

You can only choose characteristic types defined as permissible on an obligation type record. Refer to *Setting Up Characteristic Types & Their Values* for more information.

Obligation Type - Algorithms

Open **Admin Menu** > **Obligation Type** and navigate to the **Algorithm** tab to define the algorithms that should be executed for obligations 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:

Warning! These algorithms are typically significant processes. The absence of an algorithm may prevent the system from operating correctly.

The following table describes each **System Event** for which you can define algorithms.

System Event	***Optional / Required***	***Description***
Bill Completion	Optional	These algorithms are executed whenever a bill is completed for an account that contains a non-canceled obligation of this type. **Note**. Algorithms of this type are called for all non-**Canceled** obligations, 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** obligations), 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.
Break Pay Plan Obligation	Optional	These algorithms are executed when a *pay plan* is manually stopped via the *pay plan maintenance page*. Click *here* to see the algorithm types available for this system event.
Cash Accounting True Up	Optional	This algorithm is responsible for making all necessary adjustments to the general

		ledger to transfer amounts from "holding" general ledger accounts to true payable accounts. It is executed when the Calculate P&I service has been called in Update mode, regardless of whether or not P&I is applicable for the obligation type. Refer to *P&I and Cash Accounting* for more information. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.
Determine Detailed Balance	Optional	This plug-in spot is responsible for breaking down an obligation's balance into balances by separate assessments and debt categories (for example: tax, penalty, interest, fees, etc). It is used in the base **P&I Calculation** plug-in and several other services that require the balance of an obligation to be broken down by assessment and debt category. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.
FT Freeze	Optional	These algorithms are executed whenever a financial transaction is frozen that is linked to an obligation of this type. Click *here* to see the algorithm types available for this system event.
Obligation Activation	Optional	These algorithms are executed when an obligation status changes from **Pending Start** to **Active**. It performs any additional activities that are necessary to activate an obligation. Click *here* to see the algorithm types available for this system event.
Obligation Cancel	Optional	These algorithms are executed when an obligation status changes to **Canceled**. It performs any additional activities that are necessary when an obligation is canceled. Click *here* to see the algorithm types available for this system event.
Obligation Creation	Optional	These algorithms are executed when an obligation is created. Click *here* to see the algorithm types available for this system event.
Obligation Information	Optional	We use the term "obligation information" to describe the basic information that appears throughout the system to describe an obligation. The data that appears in "obligation information" may be constructed using an algorithm plugged in here.

		Refer to *Defining Obligation Information* for more information on when this algorithm is used. Click *here* to see the algorithm types available for this system event.
Obligation Stop	Optional	These algorithms are executed whenever an obligation's status changes from **pending stop** to **stopped**. Click *here* to see the algorithm types available for this system event.
Obligation Stop Initiation	Optional	These algorithms are executed whenever an obligation's status becomes **pending stop**. Click *here* to see the algorithm types available for this system event.
P&I Calculation	Optional	This plug-in spot is responsible for *calculating penalty and interest* for the input obligation. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.
P&I Eligibility	Optional	This plug-in spot is invoked by the base Calculate Penalty and Interest plug-in. It receives an obligation id as input and returns an indication of whether the obligation is eligible or ineligible for P&I. Refer to *P&I Eligibility* for more information. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.
P&I Post Processing	Optional	This plug-in spot is invoked by the base P&I Calculation plug-in after all the penalty and interest calculations are finished. Algorithms plugged into this plug-in spot are responsible for additional logic performed at the end of the P&I process. Refer to *P&I Post Processing* for more information. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.
P&I Pre-processing	Optional	This plug-in spot is invoked by the base P&I Calculation plug-in and is used to populate information needed during P&I calculations. Refer to *P&I Pre-Processing* for more information. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

P&I Prepare Periodic Balance	Optional	This plug-in spot is called from the Calculate Penalty and Interest plug-in spot for each date range being calculated. It is responsible for passing the appropriate FTs to the **Determine Detailed Balance** plug-in for the current date range. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.
Payment Freeze	Optional	These algorithms are executed whenever a payment is frozen. Click *here* to see the algorithm types available for this system event.
Process Pay Plan Scheduled Payment	Optional	These algorithms are executed by the Pay Plan Scheduled Payment Processing background process whenever a scheduled payment is due. If the pay plan obligation is unmonitored, this algorithm is not called. This algorithm should be specified on pay plan obligation types to create the necessary adjustments for the pay plan obligation. Click *here* to see the algorithm types available for this system event.
Retrieve FT List	Optional	This plug-in spot is responsible for retrieving the existing FTs for an obligation. It obtains all the data for each FT that is needed for **P&I Calculation** logic and the **Determine Detailed Balance** logic. Refer to the *Big Picture of Penalty and Interest* for more information. Click *here* to see the algorithm types available for this system event and for more information about the behavior of this plug-in spot.

Obligation Type - Pay Plan Recommendation Rule

Open **Admin Menu, Obligation Type** and navigate to the **Pay Plan Rec'n Rule** tab to define the recommendation rules that are valid for pay plan obligations of this type.

Description of Page

If the obligation type's special role is **Pay Plan**, you may define the **Recommendation Rules** that are valid on pay plan obligations of this type. Check the **Use As Default** box to indicate the default recommendation rule for obligations of this type.

Fastpath:

For more information about pay plans, refer to *Defining Pay Plan Options*.

Where Used

The pay plan recommendation rules are used to recommend the payment amount and payment schedule for pay plan obligations. Refer to *Maintaining Pay Plans* for more information.

Obligation Type - P&I Control

Open **Admin Menu, Obligation Type** and navigate to the **P&I Control** tab to define the penalty and interest control record that governs the *penalty and interest calculation* rules for obligations of this type.

Description of Page

The P&I Control and its P&I Rules define the configuration governing the tax authority's penalty and interest calculations. As these rules change over time, a new P&I Control record with the new applicable P&I Rules is created and linked to the applicable obligation types for the effective date of the change.

The following fields display:

- **Effective Date** is the date the P&I Control rate becomes effective.
- **P&I Control** defines the P&I rules used to calculate the individual penalty and interest charges.

Note:

P&I configuration. Refer to *Setting Up Penalty and Interest Options* for more detail about the configuration required for calculating penalty and interest.

Where Used

This information is used by the P&I Calculation plug-in to apply specific rules.

Obligation Type - BC Template

Open **Admin Menu, Obligation Type** and navigate to the **BC Template** tab to define the billable charge templates that can be used on obligations of a given type.

Note:

Only billable charges have billable charge templates. Only obligations 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 obligation 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 obligations of this type (if any).

Where Used

This information is used to limit the billable charge templates that can be used for a given obligation type.

Obligation Type - BC Upload 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 obligation's obligation 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, Obligation Type** and navigate to the **BC Upload Override** tab to define override values for a given Obligation 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 **Obligation 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.

Setting Up Terms and Conditions

Your obligation type start options can reference terms and conditions (T&C's) that should be defaulted onto new obligations. Each T&C is identified with a terms and condition code. To define a terms and conditions code, open **Admin Menu** > **Terms and Conditions**.

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 Obligation Type Start Options

Obligation type start options save users time and prevent data entry errors because they default many values on an obligation (e.g., the rate schedule, recurring charge amount, contract riders, contract terms, characteristics, terms and conditions, etc. can all be defaulted onto an obligation from a start option).

An obligation type may have zero or more start options.

- An obligation type without start options is usually one that has a very limited number of options. For example, if an obligation type has a single valid rate and no taxpayer-specific contract values, you don't need to setup a start option (the obligation's default rate can default based on the information defined when you setup the obligation type).
- An obligation type with multiple start options is one where many different permutations are possible. For example, an obligation type that can have multiple rates and each rate can have multiple riders is a good candidate for start options (where each start option will default, for example, a specific rate and set of contract riders).

A start option's default values may change over time (i.e., the information on a start option is effective-dated).

Start options can cause a great deal of information to be populated on an obligation. There are several ways to change this default information:

- A user may override this information using the *obligation* transaction.
- If the obligation is in **active** or **pending stop** states, a button appears on *Obligation - Main* called **Apply New Start Option**. When pressed, the user is allowed to define a start option and the date its terms become effective on the obligation. Refer to *Changing A Start Option* for the details of this functionality.

The topics in this section describe how to setup start options.

Note:

The merge transaction can save setup time. The Obligation Type Start Options Merge transaction can be used to construct a start option by copying pieces from other start options.

Contents

Obligation Type Start Option - Main

Open **Admin Menu, Obligation Type Start Option** and navigate to the **Main** tab to define an obligation type's start options.

Description of Page

Every start option is uniquely identified by the following fields:

Division & Obligation Type	Enter the division and obligation type to which the start option is linked.
Start Option	Enter the unique identifier of the option. Pick something easy to recognize as this will be used by CSRs to pick an option when they start service.
Effective Date	Enter the earliest effective date. It should be the same as the earliest effective date of the start option's rate (although it doesn't hurt for it to be earlier). The date defaults to the current date. (The status, below, should be **Active** .)

The remaining fields further describe a start option.

Enter a **Description** for the start option.

Indicate its **Status**. For new start options, the status should be **Active**. When it's no longer applicable, change it to **Inactive**.

Enter the **Rate Schedule** that should be defaulted onto obligations created using this option.

Note:

Only rates that meet the following criteria may be specified: 1) the rate must be defined as valid for the division / obligation type, and 2) the rate must have at least one **Finished** rate version.

Fastpath:

For more information about an obligation's rates, refer to *Obligation - Rate Info*.

Enter the **Recurring Charge Amount** that should be defaulted onto obligations created using this option. This field is only visible when the obligation type allows recurring charges. In addition, the prompt for this field is defined on the

obligation type table on the billing window (e.g., it could appear as Payment Amount, Budget Amount, or Installment Amount).

Enter the **Currency Code** in which monetary amounts are denominated.

Note:

Default Note. The currency code defaults from the *installation record*.

Enter the **Total Amount to Bill** that should be defaulted onto obligations created using this option. The prompt for this field is defined on the obligation type table on the billing window.

Use **Number of Payment Periods** is not applicable in this release.

If your obligation type has a special role of **Billable Charge**, then you can setup the start option to automatically create a billable charge when an obligation is created using this start option. To use this feature you should turn on the **Create Billable Charge** switch and specify the **Billable Charge Template** that will be used to create the billable charge. These fields are only allowed for obligation types with special role of **Billable Charge**.

Fastpath:

Refer to *Setting Up Billable Charge Templates* for more information about templates.

Obligation Type Start Option - Rate Info

Open **Admin Menu, Obligation Type Start Option** and navigate to the **Rate Info** tab to define the start option's default values for contract riders and contract values.

Description of Page

The information in the **Contract Riders** collection defines the contract riders to be defaulted onto obligations created using this start option. The following fields are required for each instance:

Rate Factor	The rate factor defines the type of rider. You may only reference rate factors designated as being applicable for contract riders.
Number of Days	The number of days the rider should be in effect. This value is used by the system to set the stop date on the obligation's contract rider. If the rider has no expiration, set this field to 0. Default note: this field will be set to 0 if left blank.

Fastpath:

For more information about a rate's contract riders, refer to *Defining General Rate Factor Information*. For more information about an obligation's contract riders, refer to *Obligation - Rate Info*.

The information in the **Contract Values** collection defines the contract values to be defaulted onto obligations created using this start option. The following fields are required for each event:

Rate Factor	The rate factor defines the type of value. You may only reference rate factors designated as allowing values in contract terms.
Number of Days	The number of days the value should be in effect. This value is used by the system to set the stop date on the obligation's contract value. If the value has no expiration, set this field to 0.

Value	The amount of the contract value.

Fastpath:

For more information about a rate's contract values, refer to *Defining General Rate Factor Information*. For more information about an obligation's contract values, refer to *Obligation - Rate Info*.

Obligation Type Start Option - Characteristics & Qty

Open **Admin Menu, Obligation Type Start Option** and navigate to the **Characteristics & Qty** tab to define the start option's default values for characteristics and contract quantities.

Description of Page

The information in the **Characteristics** collection defines the characteristics to be defaulted onto obligations created using this start option. The following fields are required for each instance:

Characteristic Type	This defines the type of characteristic. Note: You may only define characteristics valid on obligations.
Characteristic Value	This defines the characteristic value that will be defaulted.

Fastpath:

For more information about an obligation's characteristics, refer to *Obligation - Chars, Qty & Rec. Charges*.

The information in the **Contract Quantity** collection defines the contract quantities to be defaulted onto obligations created using this start option. The following fields are required for each instance:

Contract Quantity Type	This defines the type of contract quantity.
Contract Quantity	The amount of the contract quantity.

Fastpath:

For more information about an obligation's contract quantities, refer to *Obligation - Chars, Qty & Rec. Charges*.

Obligation Type Start Option - Terms and Conditions

Obligations may contain legal terms and conditions (T&Cs) that govern the agreement the taxpayer has with the tax authority.

The system defaults T&Cs for an obligation from the start option used to initially create it, if applicable. You can change an obligation's T&Cs at will.

Open **Admin Menu, Obligation Type Start Option** and navigate to the **Terms and Conditions** tab to define the start option's default terms and conditions.

Description of Page

The information in the grid defines the terms and conditions to be defaulted onto obligations created using this start option. The following fields are required for each instance:

Terms and Conditions	This is the code that identifies a term and condition (T&C).

Number of Days	The number of days the T&C should be in effect. This value is used by the system to set the end date on the obligation's T&C. If the T&C has no expiration, set this field to 0. This field is defaulted to 0 if left blank.

Obligation Type Start Options Merge

Use this page to modify an existing obligation type start option (start option) by copying information from other start options.

Note:

Note. The target start option must exist prior to using this page. If you are creating a new start option, you must first go to the Start Option page to add the new start option and then navigate to the merge page to copy collection information.

Note:

Duplicate versus Merge. The Start Option page has *Duplication* capability. You would duplicate a start option if you want to a) create a new start option AND b) populate it with all the information from an existing start option. You would use the start option merge page if you want to build a start option using pieces of one or more start options.

Start Options Merge - Main

Open **Admin Menu, Obligation Type Start Options Merge** and navigate to the **Main** tab to open this page.

Description of Page

Select the **Original Start Option**, which is the target for merging the start option collection information.

Select the **Merge From Start Option**, which is your template start option to copy the collections from.

Note:

Note. You may only copy information from one Merge From start option at a time. If you wish to copy information from more than one start option, select the first Merge From start option, copy the desired records, Save, then select the next Merge From start option.

The left portion of the page will display any existing records in the collections for the original start option. The right portion of the page will display the existing records in the collections for the Merge From start option.

You may use the **Copy All** button to copy all the records in all the collections from the Merge From start option to the Original start option. If you do not choose to copy all, you may copy records individually as described below.

The left portion of the **Contract Riders** collection initially displays existing contract riders linked to the original start option. In the **Merge Type**, you will see the word **Original**, for any of these records. The **Rate Factor** and **Number of Days** for each contract rider are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The left portion of the **Contract Values** collection initially displays existing contract values linked to the original start option. In the **Merge Type**, you will see the word **Original**, for any of these records. The **Rate Factor**, **Number of Days** and **Value** for each contract value is displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The topics, which follow, describe how to perform common maintenance tasks:

Removing A Row From A Grid

If you wish to remove a record linked to the Original start option, click the "-" button to the left of the record.

Adding A New Row To A Start Option

You may move any of the records from the Merge From start option to the original start option by selecting the left arrow adjacent to the desired row. Once a record is moved it will disappear from the Merge From information and appear in the Original information with the word **Merge** in the Merge Type column.

The screen will redisplay with the characteristic moved to the left portion of the page.

Removing An Uncommitted Row From A Start Option

If you have copied a row across by mistake, you may remove it by clicking on the right arrow adjacent to the appropriate record.

Fastpath:

Refer to *Editable Grid* in the system wide standards documentation for more information about adding records to a collection by selecting from a list.

Start Options Merge - Characteristics and Quantities

Open **Admin Menu, Obligation Type Start Options Merge** and navigate to the **Characteristics and Quantities** tab to copy rows in the characteristic and contract quantity collections.

Description of Page

The left portion of the **Characteristics** collection initially displays existing characteristics linked to the original start option. In the **Merge Type**, you will see the word **Original**, for any of these records. The **Characteristic Type** and **Characteristic Value** for each characteristic are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

The left portion of the **Contract Quantity** collection initially displays existing contract quantities linked to the original start option. In the **Merge Type**, you will see the word **Original**, for any of these records. The **Contract Quantity Type** and **Contract Quantity** for each contract quantity are displayed. In the right portion of the collection, the existing records in the merge from start option are displayed initially.

Fastpath:

Refer to Obligation Type Start Options Merge - Main for more information about how to perform common maintenance tasks for the grids displayed on this tab page.

Start Options Merge - Terms and Conditions

Open **Admin Menu, Obligation Type Start Options Merge** and navigate to the **Terms and Conditions** tab to copy terms and conditions (T&Cs).

Description of Page

The left side of the **Terms and Conditions** grid initially displays the T&Cs linked to the original start option. On the right side, the T&Cs linked to the merge from start option are displayed initially.

Background Processes Addendum

This chapter is an addendum to the general *Defining Background Processes* chapter. This addendum describes the background processes that are provided with Oracle Enterprise Taxation Management.

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 payment upload process creates payments for every record that is pending
- The overdue event monitor activates pending overdue events that have reached their trigger date

Processes of this type tend to use a business date in their determination of what's ready. 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*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate 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 background process uses the information on this staging table to create the flat file that is used to interface information	Yes	*MAX-ERRORS*	Yes	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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.				
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	Yes	*MAX-ERRORS*	Yes	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		are created. This is handled by the complementary background process **APAYDSFR**. Refer to *Installation Options - Billing* and *Automatic Payments* for more information.				
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*	Yes	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate 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 the APAYACH. Refer to *Creating Automatic Payment Tender Controls* for more information.	No	*MAX-ERRORS*	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*	No	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate 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*	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. 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	No	VERIFY-ONLY-SW *MAX-ERRORS*	No	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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). - 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				

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		process with the switch set to "Y"). Refer to *The Big Picture of Balance Control* for more information.				
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*	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*	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*	Yes	100/15
C1-ADMOV	CIPLOVMB	The overdue monitor uses	Yes	*MAX-ERRORS*	Yes	2000/15

Batch Control ID	Program Name	Descripti	Multiple Threads	Extra Paramete	Error Generate To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		your overdue rules to collect overdue debt. Refer to *How Does The Overdue Monitor Work?* for more information.				
C1-ADUP1	Java	This is the first of two background processes that load the contents of the adjustment upload staging records into the various adjustment tables. The background process performs high level validation and identifies the obligation for each of the adjustment upload staging records. If any errors are found, the status of the Adjustment Staging Control is set to **Error** . If no errors are found, the status of the Adjustment Staging Control is set to **In Progress**. Refer to *Interfacing Adjustments* for more information.	Yes	ADJ-STG-CTL-ERR-TD-TYPE (To Do Type for Staging Control Errors) ADJ-STG-CTL-ERR-TD-ROLE (To Do Role for Staging Control Errors) ADJ-UPL-STG-ERR-TD-TYPE (To Do Type for Adj Staging Errors) ADJ-UPL-STG-ERR-TD-ROLE (To Do Role for Adj Staging Errors) *MAX-ERRORS*	Yes	300/15
C1-ADUP2	Java	This is the second of two background processes that load the contents of the adjustment	Yes	*MAX-ERRORS*	Yes	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		upload staging records into the various adjustment tables. This process creates adjustments for adjustment upload staging records with a **Pending** status where the Obligation ID is populated. The upload staging status is changed to **Complete**. Refer to *Interfacing Adjustments* for more information.				
C1-ADURS	Java	This process picks up all adjustment upload staging records that are In Suspense. For each record, it calls the 'Resolve Suspense' plug-in for the adjustment upload staging's adjustment type. It's the algorithm's responsibility for performing the appropriate logic to resolve the suspense. The background process can be run periodically according to the implementation's requirements. Refer to *Interfacing Adjustments*	Yes	*MAX-ERRORS*	Yes	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		for more information.				
C1-APYDF	Java	This process distributes and freezes automatic payments using payment distribution rules. The background process reads all the automatic payment records whose scheduled distribution date is on or before the business date. For each record, payment(s) and pay segment(s) are created using the payment event's distribution rule(s). Payments are frozen.	Yes	*MAX-ERRORS*	Yes	200/NA
C1-CALPI	Java	This process brings P&I up to date to the input business date for all obligations. Refer to Refer to *The Big Picture of Penalty and Interest* for more information.	Yes	*MAX-ERRORS*	Yes	200/NA
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	Yes	NEXT-STATUS-CD (Next Status Code) NEXT-TR-COND-FLG (Next Transition Condition) *MAX-ERRORS*	Yes	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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.				
C1-FBTC	Java	This process balances the tender controls associated with processed form batch header records. It is run at the very end of the upload process when all of the following are true: a) The form batch headers are in a completed state. b) All the form upload staging records in each batch are in some final state - e.g. form added, etc. c) The tax or registration forms created from the upload records are in some final state or are suspended. (Any payments would have been	Yes	restrictToFrmBatchHea (Restrict By Form Batch Header ID) *maxErrors*	HeaderID	10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		created at any of these states.)				
C1-ODET	CIPLOETB	The overdue / cut event manager activates all overdue and cut 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 or cut 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*	Yes	2000/15
C1-PEPL1	CIPPEL1B	This is the first of three	Yes	*MAX-ERRORS*	Yes	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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**. Refer to *Interfacing Payments Using Distribution Rules* for more information.					
C1-PEPL2	CIPPEL2B	This is the second of three background	Yes	*MAX-ERRORS*	Yes	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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.				
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*	No	300/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
C1-PAYPA	CIPGACRB	The Pay Plan scheduled payment autopay create process creates autopay records for any scheduled pay plan payments where the account is set up for autopay and the pay plan is not excluded from autopay. For more information about pay plan auto payments, refer to *Automatic Payment and Pay Plans*.	Yes	*MAX-ERRORS*	Yes	300/15
C1-PAYPS	CIPGNPSB	The pay plan scheduled payment process performs processing for scheduled payment records with a payment date on or before the process business date. After processing, the scheduled payment status is updated to processed. For more information, refer to *Processing Scheduled Payments*.	Yes	*MAX-ERRORS*	Yes	300/15
C1-RDAMT	CIPFFTRB	This process looks for financial transactions linked to each obligation that are older than	Yes	*MAX-ERRORS*	Yes	100/10

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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 obligation's balance.				
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 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 Status Code) *MAX-ERRORS*	Yes	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
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*	Yes	200/15
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	Yes	*MAX-ERRORS*	No	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		journal entries that are interfaced to your general ledger. Refer to *The GL Interface* for more information.				
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*	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 obligation exists. Refer to *Resolving Exceptions Automatically* for more information.	Yes	*MAX-ERRORS*	No	200/15
SAACT	CIPCSATB	The obligation activation process updates pending start	Yes	*MAX-ERRORS*	Yes	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		and pending stop obligations.				

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Monitor Processes

A periodic monitor batch process is provided for any maintenance object whose business object defines a *lifecycle*. In addition deferred monitor batch process is provided if a business object supplied in the base product required a deferred process for one of its states.

The following table highlights some important monitor processes.

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
C1-FBHM	Java	This process invokes monitoring rules associated with the current state of form batch header. By default, the process periodically monitors form batch headers whose current state is not associated with a batch code. Refer to *Forms Upload Process Flow* for more information.	Yes	maintenanceObject (Maintenance Object) isRestrictedByBatchCode (Restrict by Batch Code) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) *maxErrors*	Yes	10/15
C1-FBHMD	Java	This process invokes monitoring rules associated with the current state of form batch header.	Yes	maintenanceObject (Maintenance Object) isRestrictedByBatchCode (Restrict by Batch Code)	Yes	10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		This batch control is set up with the **Restrict By Batch Code**parameter set to true to restrict processing to form batch headers whose current state is associated with this specific batch control. Refer to *Forms Upload Process Flow* for more information.		restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) *maxErrors*		
C1-FUSDM	Java	This process invokes monitoring rules associated with the current state of forms for form upload staging where the state of its form batch header is **In Progress**. This is for ensuring that specific form upload staging processing only happens when the form batch header has passed validation. This batch control is set up with the **Restrict By Batch Code**parameter set to true to restrict processing to form batch headers whose current state is associated with this specific batch control. Refer to *Forms Upload Process*	Yes	isRestrictedByBatchCode (Restrict by Batch Code) restrictToType (Restrict by Form Upload Staging Type) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch Header Status Code) *maxErrors*	Y Code	10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		Flow for more information.				
C1-FUSM	Java	This process invokes monitoring rules associated with the current state of forms for form upload staging. By default, the process periodically monitors forms whose current state is not associated with a batch code. Batch parameters govern whether the processing is further restricted by batch code, form type, business object and status. Refer to *Forms Upload Process Flow* for more information.	Yes	isRestrictedByBatchCo (Restrict by Batch Code) restrictToType (Restrict by Form Upload Staging Type) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch Header Status Code) *maxErrors*	YCode	10/15
C1-FUSPC	Java	This process invokes monitoring rules associated with the current state of forms for form upload staging where the state of its form batch header is **Pending Complete**. This is for ensuring that specific form upload staging processing only happens when the form batch header has passed validation. The batch code is set up with the **Restrict by**	Yes	isRestrictedByBatchCo (Restrict by Batch Code) restrictToType (Restrict by Form Upload Staging Type) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch	YCode	10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		Batch Code parameter set to true to restrict processing to forms whose current state is associated with the specific batch control. Refer to *Forms Upload Process Flow* for more information.		Header Status Code) *maxErrors*		
C1-RGFBD	Java	This batch process invokes monitoring rules associated with the current state of registration forms. The batch code is set up with the **Restrict by Batch Code** parameter set to true to restrict processing to registration forms whose current state is associated with the specific batch control. The optional parameters **Form Batch Header ID** and **Form Batch Header Status** allow for specific monitoring of uploaded registration forms. Refer to *Forms Upload Process Flow* for more information.	Yes	isRestrictedByBatchCode (Restrict by Batch Code) restrictToType (Restrict by Form Type) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch Header Status Code) threadBy (Thread by Flag) *maxErrors*	Yes	10/15
C1-RGFBM	Java	This batch process invokes monitoring rules associated with the current state of registration forms. By	Yes	isRestrictedByBatchCode (Restrict by Batch Code) restrictToType (Restrict by Form Type)	Yes	10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		default, the process periodically monitors registration forms whose current state is not associated with a batch code. Batch parameters govern whether the processing is further restricted by batch code, form type, business object, form status, form batch header and form batch header status. The optional parameters **Form Batch Header ID** and **Form Batch Header Status** allow for specific monitoring of uploaded registration forms. Refer to *Forms Upload Process Flow* for more information.		restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch Header Status Code) threadBy (Thread by Flag) *maxErrors*		
C1-TXMTD	Java	This batch process invokes monitoring rules associated with the current state of tax forms. he batch code is set up with the **Restrict by Batch Code** parameter set to true to restrict processing to tax forms whose current state is associated with the specific batch control.	Yes	isRestrictedByBatchCode (Restrict by Batch Code) restrictToType (Restrict by Form Type) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by	Yes	10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		The optional parameters **Form Batch Header ID** and **Form Batch Header Status** allow for specific monitoring of uploaded tax forms. Refer to *Forms Upload Process Flow* for more information.		Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch Header Status Code) threadBy (Thread by Flag) *maxErrors*		
C1-C1TXMTR	Java	This batch process invokes monitoring rules associated with the current state of tax forms. By default, the process periodically monitors tax forms whose current state is not associated with a batch code. Batch parameters govern whether the processing is further restricted by batch code, form type, business object, status, form batch header and form batch header status. The optional parameters **Form Batch Header ID** and **Form Batch Header Status** allow for specific monitoring of uploaded tax forms. Refer to *Forms Upload Process*	Yes	isRestrictedByBatchCode (Restrict by Batch Code) restrictToType (Restrict by Form Type) restrictToBusinessObject (Restrict by Business Object) restrictToBOStatus (Restrict by Status Code) restrictToFrmBatchHeaderID (Restrict by Form Batch Header ID) restrictToFrmBatchHeaderStatus (Restrict by Form Batch Header Status Code) threadBy (Thread by Flag) *maxErrors*		10/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		Flow for more information.				

Please refer to *Column Descriptions* for more information on the columns used in the table above.

Fastpath:

Refer to *Monitoring Batch Processes* for more information.

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.

Note:

Business Intelligence Extracts. If you are using Oracle Utilities Business Intelligence, there will be many other extract processes. These additional processes are responsible for extracting the data sent to the data-warehouse. Please see the product's fact and dimension chapter of the business intelligence documentation for a description of each extract 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	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed *MAX-ERRORS*	NA

Batch Control ID	***Program Name***	***Description***	***Multiple Threads***	***Extra Parameters***	***Records Between Commits / Minutes Between Cursor Re-Initiation***
		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.			
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 product program to your specific needs, please following the standard steps: • Copy the base product program to your own program. Your own program should be prefixed with the letters **CM** (which stands for "taxpayer modification"). This is important as it prevents the upgrade process from	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed *MAX-ERRORS*	NA

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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. 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.	No	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed *MAX-ERRORS*	NA

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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. In order to adapt the base product program to your specific needs, please following the standard steps: • Copy the base product program to your own program. Your own program should be prefixed with the letters **CM** (which stands for "taxpayer 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 *The Big Picture of*			

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		Collection Agency Referrals for more information.			
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 product program to your specific needs, please following the standard steps: • Copy the base product program to your own program. Your own program should be prefixed with the letters **CM** (which stands for "taxpayer 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.	No	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed *MAX-ERRORS*	NA

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		Refer to *The GL Interface* for more information.			
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	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*	NA

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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. 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 template that expects this type of format. 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.			
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	Yes	*FILE-PATH*= directory path into which output should be placed FILE-NAME= name of file into which output should be placed *MAX-ERRORS*	NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		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. The **FILE-PATH** parameter controls where the output files are placed. Refer to *Printing Bills* for more information.			

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	Descripti	Multiple Threads	Extra Paramete	Error Generate 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	No	*MAX-ERRORS*	No	NA

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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.				
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*	No	NA
F1-AVTBL	Java	This process regenerates table and column information to be displayed by the Application Viewer. It reads the meta-	No	*MAX-ERRORS*	No	NA

Batch Control ID	Program Name	Descripti	Multiple Threads	Extra Paramete	Error Generate To Do	Records Between Commits / Minutes Between Cursor Re-Initiation
		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.				
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*	Yes	100/15
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*	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	Yes	200/15

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		Complete for more information.		the bills were created. This should be entered in the format YYYY-MM-DD. *MAX-ERRORS*		
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*	No	200/15
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	No	BATCH-CD-IN-PROGRESS = the batch control ID of the abnormally terminated batch process BATCH-NBR-IN-PROGRESS	No	N/A

Batch Control ID	*Program Name*	*Descripti*	*Multiple Threads*	*Extra Paramete*	*Error Generate To Do*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		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.		= 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. 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*		

Please refer to *Column Descriptions* for more information on the columns used in the table above.

ToDo Entry Processes

These are background processes whose main purpose is to generate To Do Entry records based on a certain condition. Refer to *ToDo 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*	200/15
TD-BIERR	CIPQBIEB	This background process creates a To Do entry for every bill that's in error.	Yes	*MAX-ERRORS*	200/15
TD-BSERR	CIPQBSEB	This background process creates a To Do entry for every bill segment that's in error.	Yes	*MAX-ERRORS*	200/15
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*	200/15
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*	200/15

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
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*	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*	200/15
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*	200/15
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*	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 obligations.	Yes	*MAX-ERRORS*	200/15
TD-PYERR	CIPQPAYB	This background process creates a To Do entry for every payment	Yes	*MAX-ERRORS*	200/15

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		that's in error or that is unfrozen.			
TD-PYUPL	CIPQPYUB	This background process creates a To Do entry for every payment staging record that's in error.	No	*MAX-ERRORS*	200/15
TD-UNBAL	CIPQPYEB	This background process creates a To Do entry for every payment event that's unbalanced.	Yes	*MAX-ERRORS*	200/15
TD-XAIUP	CIPQXAUB	This background process creates a To Do entry for every XAI Upload Staging in error.	Yes	*MAX-ERRORS*	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. 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.

Fastpath:

Refer to *Validate Information In The Staging Tables* for more information about these processes and where their errors appear.

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.	200/15
VAL-APPL	CIPVAPPB	Validate the appeal 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.	200/15
VAL-AUDT	CIPVAUDB	Validate the audit case 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.	200/15
VAL-BCHG	CIPVBCGB	Validate the billable charge object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value	200/15

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
				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*	
VAL-BKCY	CIPVBKCB	Validate the bankruptcy 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.	200/15
VAL-CASE	CIPVCSEB	Validate the case 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.	200/15
VAL-CLCS	CIPVCCSB	Validate the collection case object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value	200/15

Batch Control ID	***Program Name***	***Description***	***Multiple Threads***	***Extra Parameters***	***Records Between Commits / Minutes Between Cursor Re-Initiation***
				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.	
VAL-OBLG	CIPVSVAB	Validate the obligation 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.	200/15
VAL-OVPY	CIPVOPPB	Validate the overpayment 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	200/15
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.	200/15

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
VAL-PREM	CIPVPRMB	Validate the location 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.	200/15
VAL-REVW	CIPVREVB	Validate the review 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.	200/15
VAL-SUPP	CIPVSUPB	Validate the suppression 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.	200/15
VAL-TAXR	CIPVTXRB	Validate the tax role object	Yes	OVRD-LOW-ID=key value to override the calculated start-key value	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
				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.	
VAL-TXFR	CIPVTXFB	Validate the tax form 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.	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 using the same program **CIPVRNVB**. These programs are typically run as part of the conversion and upgrade processes.

In all cases, the processes are not multi-threaded and do not include extra parameters. In addition, Records Between Commits and Minutes Between Cursor Re-Initiation are not applicable.

Fastpath:

Refer to *Validate Information In The Staging Tables* for more information about these processes and where their errors appear.

Batch Control ID	Description
CIPVAAPV	Foreign Key validation for Account Automatic Payment
CIPVACHV	Foreign Key validation for Account Characteristics
CIPVACPV	Foreign Key validation for Account Person Relationship

Batch Control ID	*Description*
CIPVADCV	Foreign Key validation for Adjustment Characteristics
CIPVADJV	Foreign Key validation for Adjustment
CIPVADLV	Foreign Key validation for Adjustment Log
CIPVADPV	Foreign Key validation for Adjustment Log Parameters
CIPVALPV	Foreign Key validation Appeal Log Parameters
CIPVAPAV	Foreign Key validation for Location Alternate Address
CIPVAPCV	Foreign Key validation for Account Person Characteristics
CIPVAPEV	Foreign Key validation for Appeal Person
CIPVAPHV	Foreign Key validation for Appeal Characteristics
CIPVAPLV	Foreign Key validation for Appeal Log
CIPVAPOV	Foreign Key validation for Appeal Related Object
CIPVAPPV	Foreign Key validation fro Appeal
CIPVAPRV	Foreign Key validation for A/P Check Request
CIPVAPSV	Foreign Key validation for Appeal Suppressed Obligation
CIPVAREV	Foreign Key validation for Appeal Review
CIPVARHV	Foreign Key validation for Collection Agency Referral History
CIPVARSI	Foreign Key validation for Credit Review Schedule
CIPVARUV	Foreign Key validation for Appeal Responsible User
CIPVAUCV	Foreign Key validation for Audit Case Characteristics
CIPVAULV	Foreign Key validation for Audit Case Log
CIPVAUMV	Foreign Key validation for Audit Case Log Parameters
CIPVAUPV	Foreign Key validation for Audit Case Related Person
CIPVAURV	Foreign Key validation for Audit Case Review
CIPVAUSV	Foreign Key validation for Audit Case Audited Obligation
CIPVAUUV	Foreign Key validation for Audit Case Responsible User
CIPVBCGV	Foreign Key validation for Billable Charge
CIPVBCHV	Foreign Key validation for Bill Characteristic

Batch Control ID	*Description*
CIPVBCLV	Foreign Key validation for Billable Charge Line
CIPVBFVV	Foreign Key validation for Rate Factor Value
CIPVBKAV	Foreign Key validation for Bankruptcy Log Parameters
CIPVBKCV	Foreign Key validation for Bankruptcy
CIPVBKHV	Foreign Key validation for Bankruptcy Characteristics
CIPVBKLV	Foreign Key validation for Bankruptcy Log
CIPVBKMV	Foreign Key validation for Bankruptcy Milestone
CIPVBKOV	Foreign Key validation for Bankruptcy Covered Obligation
CIPVBKPV	Foreign Key validation for Bankruptcy Related Person
CIPVBKUV	Foreign Key validation for Bankruptcy Responsible User
CIPVBLLV	Foreign Key validation for Bill Header
CIPVBLMV	Foreign Key validation for Bill Messages
CIPVBLRV	Foreign Key validation for Bill Routing
CIPVBSAV	Foreign Key validation for Bill - Obligation Balance Snapshot
CIPVBSCV	Foreign Key validation for Bill Segment Calc Header
CIPVBSLV	Foreign Key validation for Bill Segment Calc Line
CIPVCARV	Foreign Key validation for Collection Agency Referral
CIPVCCCV	Foreign Key validation for Collection Case Characteristics
CIPVCCHV	Foreign Key validation for Case Characteristics
CIPVCCLV	Foreign Key Validation for Collection Case Log
CIPVCCOV	Foreign Key Validation for Collection Case Overdue Process
CIPVCCPV	Foreign Key Validation for Collection Case Log Parameter
CIPVCCSV	Foreign Key Validation for Collection Case
CIPVCLCV	Foreign Key validation for Adjustment Calc Line Characteristics
CIPVCLGV	Foreign Key validation for Case Log
CIPVCPAV	Foreign Key validation for Case Log Parameters
CIPVCRTV	Foreign Key validation for Credit Rating History

Batch Control ID	*Description*
CIPVCSCV	Foreign Key validation for Customer Contact
CIPVFTCV	Foreign Key validation for Financial Transaction Characteristics
CIPVFTFV	Foreign Key validation for Financial Transaction
CIPVFTGV	Foreign Key validation for Financial Transaction Gen Ledger
CIPVFTPV	Foreign Key validation for Financial Transaction Process
CIPVMSGV	Foreign Key validation for Account Bill Messages
CIPVNBSV	Foreign Key validation for Pay Plan / Obligation
CIPVNPMV	Foreign Key validation for Pay Plan Payment Schedule Parameter Values
CIPVNSPV	Foreign Key validation for Pay Plan Scheduled Payments
CIPVOPCV	Foreign Key validation for Overpayment Process Characteristics
CIPVOPLV	Foreign Key validation for Overpayment Process Log
CIPVOPMV	Foreign Key validation for Overpayment Process Log Parameter
CIPVOPPV	Foreign Key validation for Overpayment Process
CIPVPAOV	Foreign Key validation for Person Address Override
CIPVPAYV	Foreign Key validation for Payment Header
CIPVPYCV	Foreign Key validation for Payment Characteristic
CIPVPCHV	Foreign Key validation for Location Characteristic
CIPVPGOV	Foreign Key validation for Location Geographic location
CIPVPIDV	Foreign Key validation for Person Identifier
CIPVPNMV	Foreign Key validation for Person Name
CIPVPPEV	Foreign Key validation for Person to Person
CIPVPPHV	Foreign Key validation for Person Phone
CIPVPRCV	Foreign Key validation for Person Characteristics
CIPVPSAV	Foreign Key validation for Person Seasonal Address
CIPVPSGV	Foreign Key validation for Payment Segment
CIPVPYCV	Foreign Key validation for Payment Characteristics
CIPVREVV	Foreign Key validation for Review

Batch Control ID	*Description*
CIPVRLPV	Foreign Key validation for Review Log Parameters
CIPVRVCV	Foreign Key validation for Review Characteristics
CIPVRVLV	Foreign Key validation for Review Log
CIPVSACV	Foreign Key validation for Obligation Characteristics
CIPVSAHV	Foreign Key validation for Obligation Rate Schedule History
CIPVSAOV	Foreign Key validation for Obligation Contract Terms
CIPVSAQV	Foreign Key validation for Obligation Contract Quantity
CIPVSARV	Foreign Key validation for Obligation Recurring Charge
CIPVSEGV	Foreign Key validation for Bill Segment
CIPVSMGV	Foreign Key validation for Obligation Message
CIPVSQTV	Foreign Key validation for Bill Segment Rate Quantity
CIPVSRRV	Foreign Key validation for Bill Segment Register Read
CIPVSUCV	Foreign Key validation for Suppression Characteristics
CIPVSUEV	Foreign Key validation for Suppression Entity
CIPVSULV	Foreign Key validation for Suppression Log
CIPVSUMV	Foreign Key validation for Suppression Log Parameters
CIPVSUPV	Foreign Key validation for Suppression
CIPVSURV	Foreign Key validation for Suppression Process
CIPVTFCV	Foreign Key Validation for Tax Form Characteristics
CIPVTFLV	Foreign Key Validation for Tax Form Log
CIPVTLPV	Foreign Key Validation for Tax Form Log Parameter
CIPVTNDV	Foreign Key validation for Payment Tender
CIPVTNCV	Foreign Key validation for Payment Tender Characteristics
CIPVTXCV	Foreign Key validation for Tax Role Characteristics
CIPVTXFV	Foreign Key validation for Tax Form
CIPVTXRV	Foreign Key validation for Tax Role

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.

Fastpath:

Refer to *The Conversion Tool* for more information about these processes and where their errors appear.

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*	200/15
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*	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.

  A separate background process exists for every table with a system-assigned key that is supported by the conversion tool. The program names of these processes are documented in *The Conversion Process* (scan for all references to "Key Assignment Program" for a matrix containing these program names).
- **Insertion Programs.** Background processes of this type insert converted rows into production from the staging database.

  A separate background process exists for every table that is supported by the conversion tool. The program names of these processes are documented in *The Conversion Process* (scan for all references to "Insertion Program" for a matrix containing these program names).

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*	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*	200/15
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*	200/15
XMLUP-PR	CIPXDXUB	Purges **completed** XML upload objects.	Yes	NO-OF-DAYS = number of days after the completion date that a **completed**	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
				XML upload object should be purged *MAX-ERRORS*	

Please refer to *Column Descriptions* for more information on the columns used in the table above.

ConfigLab Processes

The following table lists system background processes used in conjunction with the *Configuration Lab*. The delivered system background processes are designed to copy sample DB processes from the demonstration database.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
CL-APPCH	CIPYUPDB	The apply changes based on compare is used to update the current environment based on the results of a compare from a **ConfigLab** or **Compare Source** environment. Refer to *Configuration Lab* for more information.	No	ENV-CODE = environment reference specified as a batch parameter on the batch run used to compare data. DB-PROC-CODE = The **Compare** DB process used to pull data from ENV-CODE. *MAX-ERRORS*	200/15
CL-COPDB	CIPYSYCB	The copy "CI_" DB processes is used to compare DB processes from the demonstration database. It is assumed that the demonstration database has been registered as a **Compare Source** environment reference. Refer to *Configuration Lab* for more information.	No	ENV-CODE = environment reference representing a registered **Compare Source** environment. STATUS-ON-ADD = default root object status when root object action is **Add**. STATUS-ON-CHANGE = default root object status when root object action is **Change**.	200/15

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
				STATUS-ON-DELETE = default root object status when root object action is **Delete**. *MAX-ERRORS*	

Archive and Purge Processes

The following table lists system background processes used in conjunction with the *Archive Engine*. Note that the DB process that represents an archive or purge procedure defines the batch control used for the first step of a purge or archive. Although, the **CIPYCPRB** program (used universally for the first step of any archive or purge procedure) is delivered with the system, there are no system batch controls that reference it. It is included in this list for clarity.

Fastpath:

For information on how to copy sample **Archive** and **Purge** DB processes from the demonstration database, refer to *How To Copy Samples From The Demonstration Database*.

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
Special - Each archive or purge procedure defines its first step's batch control.	CIPYCPRB	The first step of any archive or purge process is to create primary archive roots objects. Note that for **Purge** DB processes, the environment reference batch parameter is optional. Refer to *Archive Engine* for more information.	Yes	ENV-CODE = environment reference of the target archive environment. TEST-MODE (Y/N) = If Y is specified, the program will write out information about primary roots to a log file.	200/15
AR-CRCHR	CIPYCRCB	The second step of any archive or purge process is to create child archive roots objects. Note that for **Purge** DB processes, the environment reference batch	Yes	ENV-CODE = environment reference of the target archive environment. DB-PROC-CODE = The archive or purge DB process that specifies the	200/15

Batch Control ID	*Program Name*	*Description*	*Multiple Threads*	*Extra Parameters*	*Records Between Commits / Minutes Between Cursor Re-Initiation*
		parameter is optional. Refer to *Archive Engine* for more information.		batch control used to create primary archive root objects (step 1 of any archive or purge). *MAX-ERRORS*	
AR-PRFK	CIPYRFKB	The third step of any archive or purge process is to check recursive foreign key relationships. Note that for **Purge** DB processes, the environment reference batch parameter is optional. Refer to *Archive Engine* for more information.	Yes	ENV-CODE = environment reference of the target archive environment. DB-PROC-CODE = The archive or purge DB process that specifies the batch control used to create primary archive root objects (step 1 of any archive or purge). *MAX-ERRORS*	
AR-DCDT	CIPYPARB	The fourth step of any archive or purge process is to move data to the target archive environment (in the case of an archive), or simply delete it (in the case of purge). Note that for **Purge** DB processes, the environment reference batch parameter is optional. Refer to *Archive Engine* for more information.	No	ENV-CODE = environment reference of the target archive environment. DB-PROC-CODE = The archive or purge DB process that specifies the batch control used to create primary archive root objects (step 1 of any archive or purge). *MAX-ERRORS*	200/15
AR-DCDTF	CIPYCARB	If desired, this background process may be run instead of *AR-DCDT* (above). This process calls the **Archive Copy Processing** algorithm specified on the DB Process Instruction to copy the data to	Yes	ENV-CODE = environment reference of the target archive environment. DB-PROC-CODE = The archive or purge DB process that specifies the batch control used to create primary	200/NA

Batch Control ID	Program Name	Description	Multiple Threads	Extra Parameters	Records Between Commits / Minutes Between Cursor Re-Initiation
		a flat file instead of an archive environment. Note that for **Purge** DB processes, the environment reference batch parameter is optional. Refer to *Archive Engine* for more information.		archive root objects (step 1 of any archive or purge). MODE = C (Copy data only), D (Delete records only), or B (Both copy and delete). The default is B (Both). *MAX-ERRORS*	

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. Refer to *Parameters Supplied to Background Processes* for more information.
- **Extra Parameters.** This column indicates if the background process uses additional parameters (in addition to those described under *Parameters Supplied to 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. Refer to *Parameters Supplied to Background Processes* for more information.

Batch Process Dependencies

The contents of this section illustrate the periodicity and dependencies between the various background processes described above.

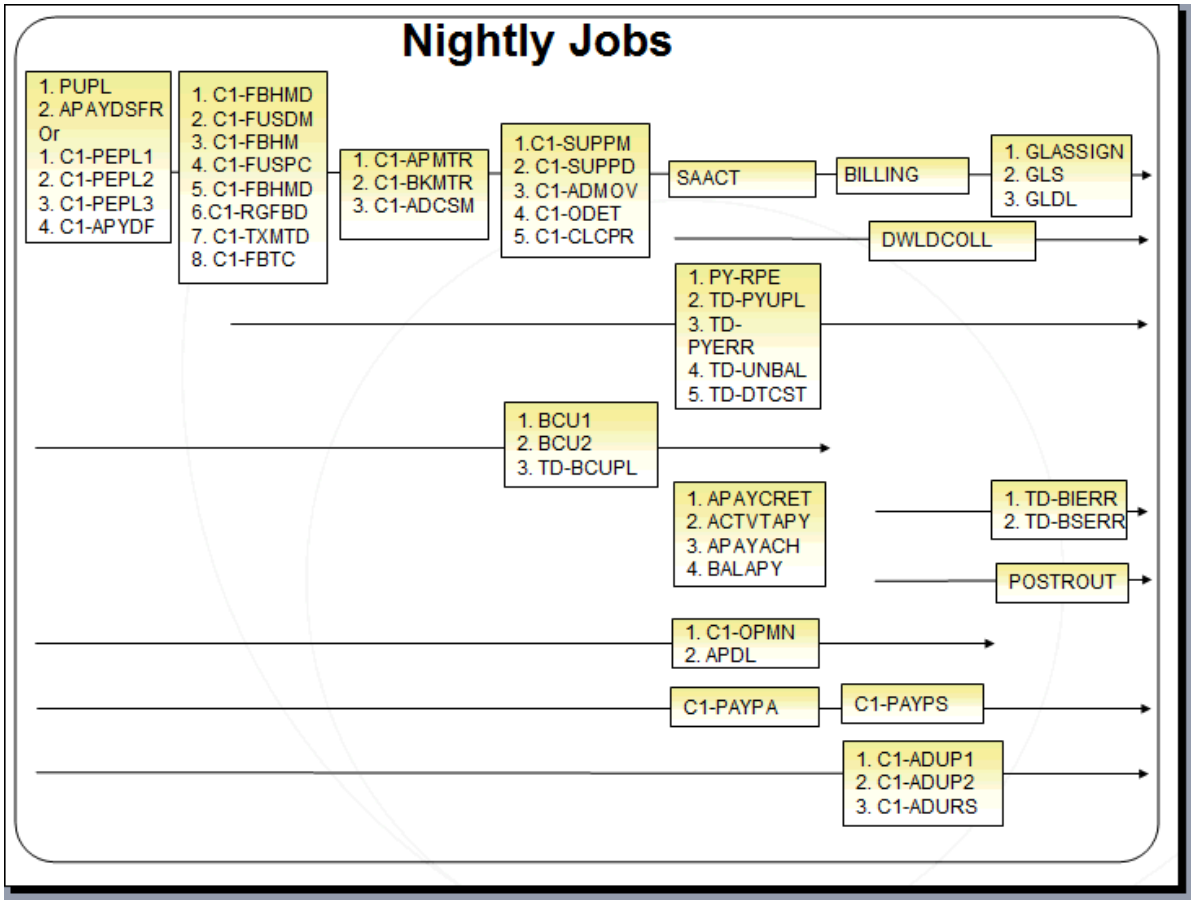
Batch Schedulers and Return Codes

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.

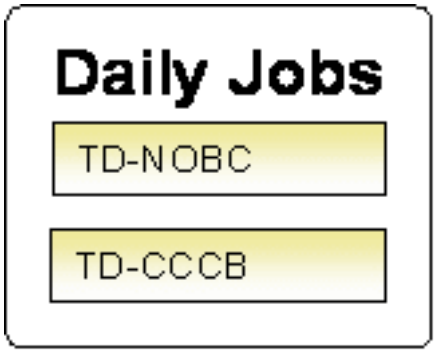
The Nightly Processes

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 first job stream's timeline indicate when the timeline's respective processes can be executed in respect of the first job stream.

The following diagram illustrates the daily batch processes for which there are no dependencies.



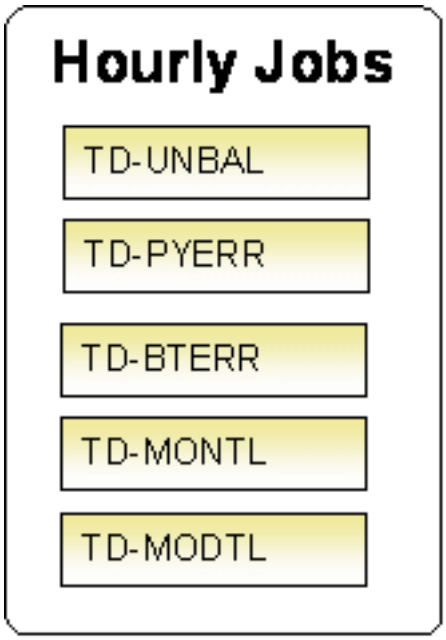
The mnemonics in the boxes refer to the individual batch processes described above.

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.



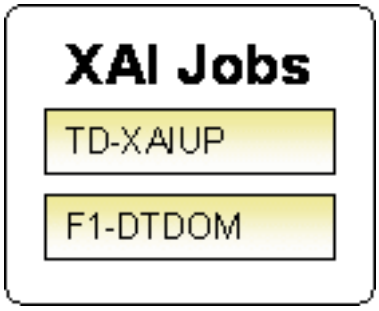
The mnemonics in the boxes refer to the individual batch processes described above. When a box contains multiple processes, these processes must be run sequentially.

Note:

No dependencies exist. As you can see, there are no dependencies between the boxes (meaning they may be run in parallel).

The XAI Processes

The following diagram illustrates the dependencies between the 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 using XAI).



These processes create and/or clean up To Do entries for XAI upload staging 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.

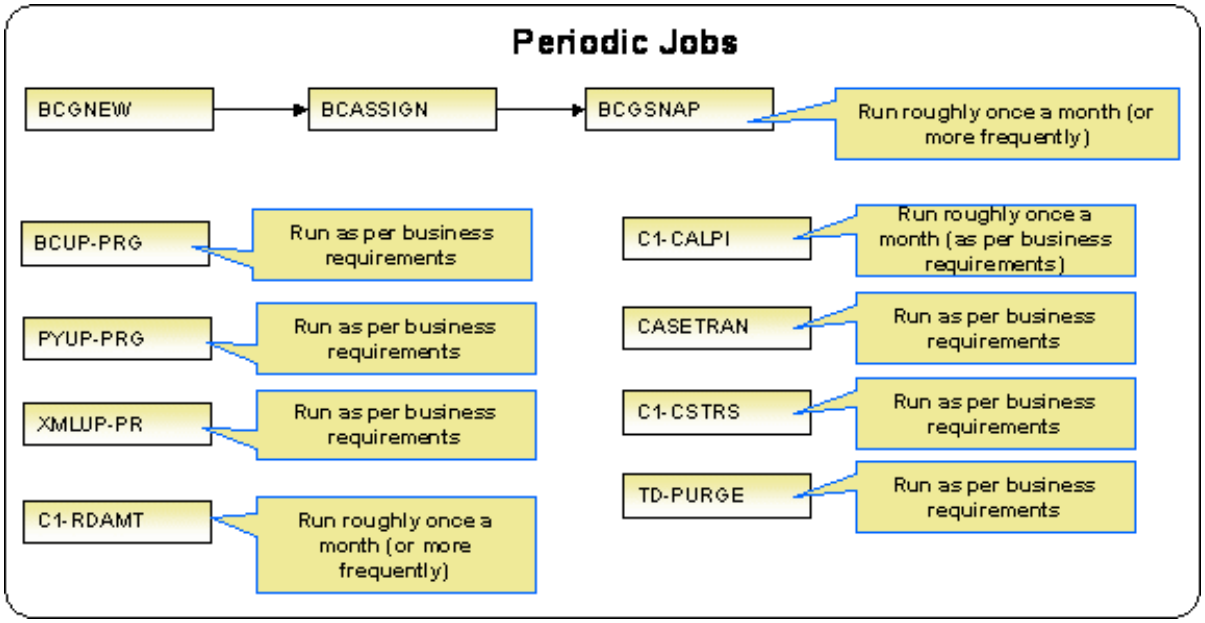
The Letter Processes

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.

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 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).



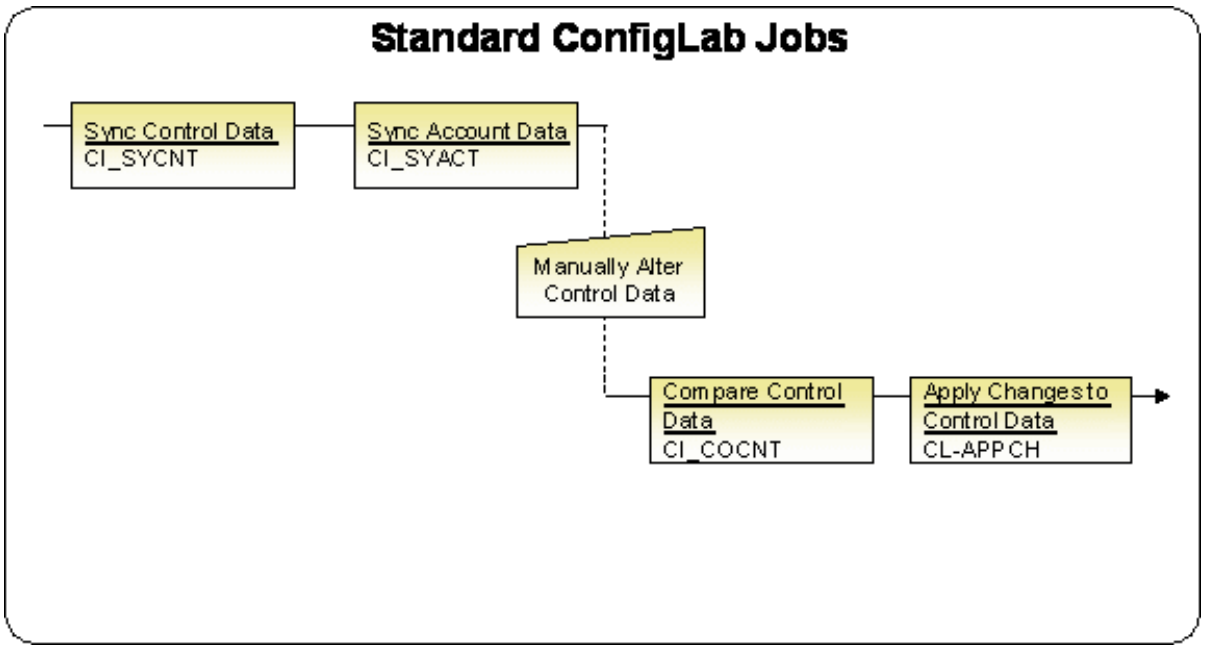
The mnemonics in the boxes refer to the individual batch processes described above.

Note:

Few dependencies exist. As you can see, there are few dependencies between the boxes (meaning they may be run in parallel).

The ConfigLab Processes

The following diagram illustrates the dependencies between the ConfigLab processes. The frequency of running ConfigLab processes is implementation specific. For more information on comparing data from an alternate environment, refer to *Configuration Lab*.

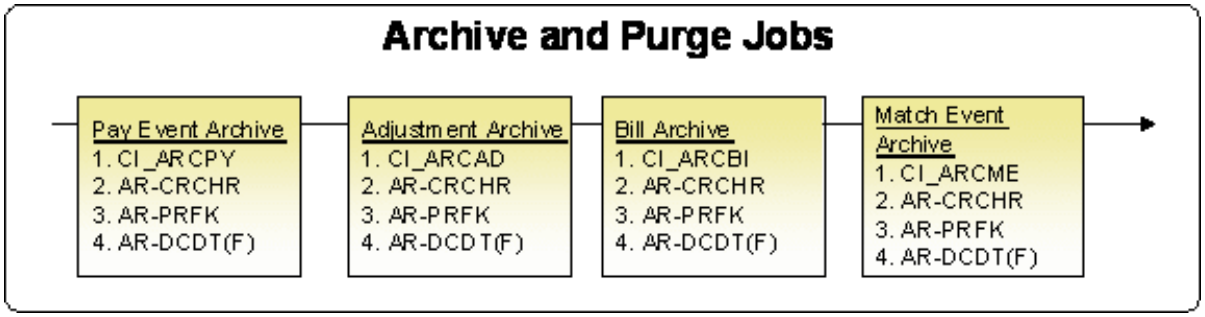


Note:

Compare Sources and Targets. The *Configuration Lab* may be used with environments other than a **ConfigLab**. In cases where control and account data are pushed to a **Compare Target**, only the top two batch processes are executed. In cases where control data is pulled from a **Compare Source** environment, only the bottom two batch processes are executed.

The Archive and Purge Processes

The following diagram illustrates the dependencies between the sample archive and purge processes. The frequency of running archive and purge processes are implementation specific. For more information on archive and purge processes, refer to *Archiving and Purging* and *Archive Engine*.



Note:

Steps 2, 3 and 4. Note that steps 2 and 3 are the same for all of the sample archive and purge jobs. For step 4, you can select from two background processes: *AR-DCDT* moves data to a target archive environment (or purges the data) and *AR-DCDTF* calls an algorithm that copies the data to a flat file. If you want to use **AR-DCDTF** for other archive jobs, you must develop your own algorithms.

How To Set Up A New Extract Processes

Several background processes delivered with the system are used to interface information out of the system. The topics in this section describe when and how to introduce an additional extract process.

Setting Up 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 taxpayer to pay via credit card or direct debit from a checking account. The **APAYACH** process delivered with the system is intended to be used to handle this function.

If you need additional automatic payment extract processes, set up the following information:

- Add a new *batch control* record. Populate the fields as follows:
- **Batch Process.** Assign an easily recognizable unique ID for the automatic payment extract process.
- **Description.** Enter a description of the automatic payment extract process.
- **Accumulate All Instances.** Turn this switch on.
- Use *Auto Pay Route Type* to define the auto pay extract process to be used for each route type.

The Big Picture of Sample & Submit

Sample and Submit refers to the ability to create Activity Requests. This is functionality that enables an implementer to design an ad-hoc batch process using the configuration tools.

Some examples of such processes are:

- Send a letter to customers that use credit cards for auto pay and the credit card expiration date is within 30 days of the current date.
- Stop auto pay for customers that use credit cards as the form of payment if the credit card has already expired. Notify the customer that their auto pay agreement has been terminated and that they need to call to reinstate.
- Select auto pay accounts that have more than X non-sufficient fund penalties, stop the auto pay agreement and notify the customer.

Note:

Note that the terms activity request and sample & submit request may be used interchangeably.

Activity Type Defines Parameters

For each type of process that your implementation wants to implement, you must configure an activity type to capture the appropriate parameters needed by the activity request.

Preview A Sample Prior To Submitting

To submit a new activity request, a user must select the appropriate activity type and enter the desired parameter values, if applicable.

After entering the parameters, the following actions are possible

- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can **Save** to submit the request, go **Back** to adjust the parameters or **Cancel** the request.

- Click **Cancel** to cancel the request.
- Click **Save** to skip the preview step and submit the request.

When an activity request is saved, the job is not immediately submitted for real time processing. The record is save in the status **Pending** and a monitor process for this record's business object is responsible for transitioning the record to **Complete**.

As long as the record is still **Pending**, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

The actual work of the activity request, such as generating customer contact records to send letters to a set of customers, is performed when transitioning to **Complete** (using an enter processing algorithm for the business object).

Credit Card Expiration Notice

The base product supplies a sample process to find customers that use credit cards for auto pay and the credit card expiration date is within x days of the current date.

To this functionality the following configuration tasks are needed:
- Define an appropriate *customer contact class* and *type* to use.
- Define appropriate activity request Cancellation Reasons. Cancellation reasons are defined using a customizable *lookup*. The lookup field name is **C1_AM_CANCEL_RSN_FLG**.
- Define an activity type for the business object **C1-NotifyExpiringCreditCardTyp**. You may define default parameter values for the number of days for expiration and customer contact class and type.

Exploring Activity Request Data Relationships

Use the following links to open the application viewer where you can explore the physical tables and data relationships behind the activity request functionality:
- Click *C1-ACM-ACTTY* to view the activity type maintenance object's tables.
- Click *C1-ACM-ACTRQ* to view the activity request maintenance object's tables.

Defining a New Activity Request

To design a new ad-hoc batch job that users can submit via Sample and Submit, first create a new Activity Type business object. The base product BO for activity type **C1-NotifyExpiringCreditCardTyp** may be used as a sample.

The business object for the activity request includes the functionality for selecting the records to process, 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 portal.

Activity Type List

The Activity Type *List zone* lists every activity type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent activity type.
- The standard actions of **Edit, Duplicate** and **Delete** are available for each activity type.
- State transition buttons are available to transition the activity type to an appropriate next state.

Click the **Add** link in the zone's title bar to add a new activity type.

Activity Type Actions

This is a standard actions zone. The Edit, Delete and Duplicate actions and appropriate state transition buttons are available.

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.

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.

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

Sample & Submit Actions

This is a standard actions zone. Use the **Edit** button to modify the parameters. Refer to *Preview A Sample Prior to Submitting* for more information.

If the activity request is in a state that has valid next states, buttons to transition to each appropriate next state are displayed.

Sample & Submit Log

This is a standard *log zone*.

Defining Bankruptcy Options

A bankruptcy is a legal procedure for dealing with taxpayer debt problems. The goal of a bankruptcy process is to relieve the debtor of some if not all of his/her debt either through liquidation of assets or readjustment of debts.

The Big Picture Of Bankruptcy

The bankruptcy process begins when the debtor files a petition.

The bankruptcy petition can include the following information:

- A list of all creditors and the amount and nature of their claims
- Source, amount and frequency of income
- List of all property
- List of monthly living expenses, such as food, clothing, shelter, utilities, taxes, transportation, medicine, etc.

  The petition date usually determines which of the debtor's obligations can be included in the bankruptcy. Any outstanding obligations as of the petition date are typically included.

Types Of Bankruptcy

Bankruptcy law typically classifies bankruptcy cases into certain types.

For instance, in the United States, bankruptcy cases are classified into 'chapters', the common ones being:

- Chapter 7 - Liquidation of assets
- Chapter 11 - Rehabilitation / reorganization for businesses
- Chapter 13 - Rehabilitation for individuals

The other chapters are:

- Chapter 9 - Bankruptcy for municipalities
- Chapter 12 - Rehabilitation for family farmers and fishermen
- Chapter 15 - Ancillary / international cases

Bankruptcy Courts

The bankruptcy petition is filed with the appropriate bankruptcy court, which assigns a case number/identifier, reviews the petition and eventually makes a ruling on the bankruptcy.

Key information about a court includes: court name, address, phone numbers, email addresses, web addresses, etc.

Bankruptcies Cover Obligations

A bankruptcy defines which of the debtor's obligations are covered.

The rules for determining eligible obligations are configurable in a plug-in spot on bankruptcy type. The base sample algorithm selects the taxpayer's obligations as of the petition date based on a specified reference date - e.g. obligation start date, obligation end date or obligation filing due date (for filing-based obligations only).

See *Setting Up the 'Determine Eligible Obligations'* for more information on how to configure this algorithm.

Other Related Persons On A Bankruptcy

Bankruptcies also identify other parties that are related to the case - e.g. attorney, trustee, administrator, etc. The information captured for each of these persons include: name, address, phone numbers, email addresses, etc.

Lifecycle Of A Bankruptcy

The steps in the bankruptcy process vary based on local bankruptcy laws.

The base product provides a bankruptcy business object **C1-Bankruptcy** that includes a number of common steps.

The bankruptcy is created in a **Pending** state that allows for all pertinent information to be set up before the bankruptcy takes effect. A **Pending** bankruptcy can be **Canceled** .

The bankruptcy goes to a **Review In Progress** step and stays in that state while the case is waiting for a decision from the court. This is the state where the bankruptcy is considered to be in effect. Therefore, any related suppressions on collection activity or penalty & interest are typically in place by this time. Proofs of claim can also be created at this point.

A bankruptcy case could result in any of the following decisions:

- Additional actions prior to discharge / dismissal - Depending in the type of bankruptcy, an agreed-upon payment plan may be established, to allow the debtor to repay some or all of his/her debt. The bankruptcy sits in an **Actions In Progress** state, while the payment plan is ongoing. The fulfillment of the payment plan typically leads to discharge. On the other hand, if the payment plan is not fulfilled, the bankruptcy could be dismissed or discharged, depending on the debtor's situation.
- **Discharge** - This action releases the debtor from personal liability from specific debts and prohibits creditors from taking any action against the debtor to collect those debts. In other words, the debt is written off. Certain types of debts may be deemed non-dischargeable, and thus, continue to be collectible. Any related suppression(s) are end-dated when discharge occurs.
- **Discharge Denial** - The case cam be denied discharge for reasons involving fraud, such as: transferring / destroying / concealing property within a certain period before or after filing bankruptcy, false oath / claim, concealed / falsified books on financial position, etc. No write offs are done in this case.
- **Dismissal** - The case can be dismissed for reasons such as failure to provide requested tax documents, failure to make payments under the confirmed plan, failure to pay administrative fees, debtor previously received a discharge for a similar bankruptcy case within a certain time period, etc. No write-offs are done in this case.

The bankruptcy can be **Closed** after all necessary post-decision tasks have been completed.

Refer to the **C1-Bankruptcy** business object metadata for an illustration of the bankruptcy lifecycle.

Creating Proofs Of Claim

A proof of claim is a written statement describing the reason(s) a debtor owes the creditor money. The creditor files this document with the court and/or the administrator of the bankruptcy case. In most cases, the proof of claim ensures that every creditor is given due consideration when resolving the debt. Some types of debts are given more priority than others. So the proof of claim is typically one of the more important documents that reviewed in bankruptcy proceedings.

The proof of claim may be deemed filed if the debt appears in the bankruptcy petition's list of creditors. Creditors can file additional proofs of claim for any debt that may not be included in the petition.

The **C1-Bankruptcy** business object provides for an ability to create proofs of claims as customer contacts. The extract logic, however, is assumed to be the implementation's responsibility.

Bankruptcies Can Cause Suppression

When obligations are included in a bankruptcy case, penalty and interest and collection activity can be suppressed for these obligations.

The type of bankruptcy determines whether or not suppression is applicable. Bankruptcy types that cause suppression should specify a suppression type.

Suppressions can be automatically created when the bankruptcy goes into a certain state. The **C1-Bankruptcy** business object has an algorithm *C1-BK-CRTSPR* that creates and activates suppression when the bankruptcy enters the **Review In Progress** state.

During the course of the bankruptcy case, obligations may get added to or removed from the bankruptcy. In some exceptional cases, the petition date may change, which may cause some obligations to become ineligible. If suppression already exists, adding / removing obligations to / from the bankruptcy will also add / remove them to / from the suppressed entities.

Suppressions can be automatically released when the debt is discharged. Your implementation will need to build your discharge processing logic (e.g. write off) based on your specific business rules. That logic can be plugged in directly on the Discharged state or configured in overdue processing.

The base product provides two algorithms that you can use, depending on the option you choose:

- *C1-BK-RLSSPR* releases the related suppression when the bankruptcy enters a certain state. Plug this in the **Discharged** state.
- *C1-OE-RLBKSU* is an overdue event activation algorithm that releases the related suppression when the overdue event is triggered.

Suppression can be automatically canceled when the bankruptcy is canceled. The **C1-Bankruptcy** business object has an algorithm *C1-BK-CNLSPR* that does this.

Refer to *The Big Picture of Suppression* for more information about suppression.

Bankruptcies Can Create Pay Plans

Payment plans can be created as a result of bankruptcy. Certain types of bankruptcy, specifically those that seek readjustment of debts typically require that the debtor fulfill an agreed-upon payment plan.

The debt is usually not discharged until the payment plan is fulfilled. In some exceptional cases, the debtor may declare hardship, in which case, the court may decide to discharge the debt anyway.

The **C1-Bankrutpcy** business object provides for an ability to create pay plans for the bankruptcy. Creating a pay plan for the bankruptcy results in the cancellation/stopping of any pay plans that existed before the bankruptcy and that cover any of the obligations that are included in the bankruptcy. It also has logic to prevent automatic discharge when the bankruptcy-related pay plans are not yet fulfilled.

Assigning Bankruptcies To Responsible Users

One or more tax authority users may be assigned as responsible users on a bankruptcy case.

A responsible user can be assigned automatically when the bankruptcy is created. The base algorithm *C1-BK-ADDUSR* assigns the user who created the bankruptcy as a responsible user. This algorithm is plugged in on **C1-Bankrutpcy** business object's **Pending** state.

Setting Up Bankruptcy Options

The following topics describe the objects that must be defined as part of bankruptcy processing setup.

Setting Up Bankruptcy Types

Bankruptcy Types contain the rules that control how bankruptcies are processed.

To set up a Bankruptcy Type, select **Admin Menu** > **Bankruptcy Type**.

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

Bankruptcy Type List

The Bankruptcy Type *List zone* lists every bankruptcy type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent bankruptcy type.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each bankruptcy type.
- Click the **Add** link in the zone's title bar to add a new bankruptcy type.

Bankruptcy Type

The Bankruptcy Type zone contains display-only information about a Bankruptcy Type. This zone appears when a Bankruptcy Type has been broadcast from the Bankruptcy Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *C1_BANKRUPTCY_TYPE*.

Setting Up The 'Determine Eligible Obligations' Algorithm

The 'Determine Eligible Obligations' algorithm suggests a list of eligible obligations during bankruptcy maintenance.

A base sample algorithm *C1-BKTY-SEOP* selects obligations as of the petition date.

To use this algorithm, open **Admin Menu** > **Algorithm**. Select the base algorithm type and configure the parameters accordingly.

Plug in the algorithm on your bankruptcy types.

Setting Up Suppression Types

Bankruptcy types that create suppression must specify a suppression type. Bankruptcy-related suppression types must specify a suppression entity level of **Obligation** .

Refer to *Suppression: Setting Up Suppression Types* for more details on setting up suppression types.

Setting Up Customer Contact Types For Proofs of Claim

To create proofs of claim using customer contacts, specify the customer contact type on the bankruptcy type. The Create Proof of Claim action on the bankruptcy will use this customer contact type.

You will need to build the extract logic for your 'proof of claim' customer contact types.

Refer to *Setting Up Customer Contact Options* for more details on customer contact types.

Setting Up Pay Plan Types And Pay Plan Recommendation Rules

Bankruptcy types that allow pay plans must specify a pay plan obligation type. In addition, if you need the system to suggest a payment schedule for bankruptcy-related pay plans, configure your pay plan recommendation rules accordingly.

Refer to *Setting Up Pay Plan Options* for more details on configuring for pay plans.

Setting Up Overdue Process Templates For Bankruptcy Discharge

If you opt to use overdue processing for discharging debt, you can specify the overdue process template for discharge on the bankruptcy type.

You can also use the following algorithms

- *C1-OE-RLBKSU* releases the related suppression when the overdue event is triggered. Plug this in your overdue process template as one of the last steps.
- *C1-BK-CRTODP* creates an overdue process when the bankruptcy enters a certain state. Plug this in the bankruptcy's Discharged state.

  Refer to *Creating Overdue Procedures* for more information on configuring overdue processing options.

Setting Up Bankruptcy Courts

Courts are set up as special types of Persons in the system. To configure court information:

- Set up a Person Type for courts. This could be a general one for all types of courts or a specific one for bankruptcy courts. You can reference the base business objects **C1-BusinessPersonType** or **C1-BusinessPerson** or your own business objects in this person type. Refer to *Defining Person Types* for more information.
- Create Person records for your bankruptcy courts, using the special Person Type you configured.

Setting Up Bankruptcy Filing Types

If you classify your bankruptcies into broad or specific categories, you can configure those categories as Bankruptcy Filing Type extendable lookup values.

- Open **Admin Menu** > **Extendable Lookup**.
- Search for and select the **Bankruptcy Filing Type** extended lookup business object.
- The list of existing bankruptcy filing types are displayed in a standard *List zone*.
- Choose an existing bankruptcy filing type to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new bankruptcy filing type.

Setting Up Bankruptcy Claim Types

If you classify obligations into specific type of claims (e.g. for 'proof of claim' purposes), you can configure those types as Bankruptcy Claim Type extendable lookup values.

- Open **Admin Menu** > **Extendable Lookup**.
- Search for and select the **Bankruptcy Claim Type** extended lookup business object.
- The list of existing bankruptcy claim types are displayed in a standard *List zone*.
- Choose an existing bankruptcy claim type to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new bankruptcy claim type.

Setting Up Milestone Types

You can define the types of milestone dates that can be captured on your bankruptcies as Milestone Type extendable lookup values.

- Open **Admin Menu** > **Extendable Lookup**.
- Search for and select the **Milestone Type** extended lookup business object.
- The list of existing milestone types are displayed in a standard *List zone*.
- Choose an existing milestone type to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new milestone type.

Setting Up Bankruptcy Relationship Types

Base values for Bankruptcy Relationship Types are supplied. You can add your specific relationship types by updating the value for the BANKRUPTCY_REL_TYPE_FLG *lookup* field.

Setting Up Assignment Roles

Base values for responsible user's Assignment Roles are supplied. You can add your specific assignment roles by updating the value for the ASSIGNMENT_ROLE_FLG *lookup* field.

Setting Up An Alert For Highlighting Open Bankruptcies

To show a Control Central alert when the person in context has any open bankruptcies, define an algorithm for the *C1-BNKR-OPN* base algorithm type and configure the alert in Installation Options.

To configure the alert, open **Admin Menu** > **Installation Options - Framework** and configure the algorithm you defined in the **Control Central Alert** system event.

Defining Audit Case Options

Tax audits are conducted to examine the accuracy of reported tax information. It could be as simple as reviewing the accuracy of the assessment on a single tax return or as complex as reviewing a number of obligations for specific filing periods.

Audits are also conducted when the tax authority finds that a taxpayer has failed to file certain tax returns that he/she is expected to file.

The Big Picture Of Audit Cases

A tax authority could have a number of reasons for initiating an audit. Internal reports and reviews usually identify specific taxpayers and/or accounts that need to be audited. The selection of taxpayers and/or accounts is based on specific criteria that the tax authority defines.

External interfaces could also send reports / information that prompt a tax authority to subject certain taxpayers to audits. The most common situation is when income information is reported for a taxpayer, who has either not filed a return or has filed but did not include that information.

Obligations Are Audited

An audit case identifies the primary account that is being audited. Any of the obligations of the primary taxpayer and other financially responsible persons on that primary account can be examined during the audit.

Other Related Persons On An Audit Case

Audit cases can also specify other parties that are related to the case. For instance, if the taxpayer is represented by an attorney, accountant, enrolled actuary, enrolled agent, paid preparer, etc., that representative is a related person on the case.

The information captured for the related persons can include: names, addresses, phone numbers, email addresses, etc.

Preparing For An Audit

An auditor may need to perform a number of tasks prior to the audit proper. These tasks can include: examining the taxpayer's information, looking at the overall picture of accounts, examining the latest tax forms, identifying any missed tax returns, creating obligations, etc.

Initial correspondence or meetings may also be conducted, to explain the audit process and request for additional documentation that will be needed during the audit.

Conducting Audits

There are three general methods of conducting audits:

Desk Audits

This method uses correspondence / letters to ask for additional information or to notify the taxpayer of a change in the assessment. In the latter case, the taxpayer responds by either sending in a payment or filing for an appeal.

Office Audits

In this method, the taxpayer or an authorized representative is asked to go to the tax authority's office and bring the necessary documentation.

Field Audits

This is considered the most aggressive method. A field audit officer goes to the taxpayer's or the authorized representative's residence or office. Tax-related information is examined on site.

Lifecycle Of An Audit Case

The steps in an audit case may vary depending on the type of audit case.

The base product provides an audit case business object **C1-AuditCase** that includes a number of common steps.

The audit case is created in a **Pending** state to allow for any preparatory tasks prior to starting the audit.

When the audit process starts, the audit case goes to the **Audit In Progress** state. If suppression is applicable, the suppression is typically created and activated at this point. The audit case stays in this state until findings are identified. The most common result of an audit is a new assessment. This happens when either an existing tax form is audited or when the auditor creates an audit form for tax returns that the taxpayer failed to file.

The findings of the audit can be subjected to any required reviews and/or approvals while the audit case is in the **Review In Progress** state. If there are multiple review steps, a separate Review process can be created from the audit case. Refer to *Creating Audit Case Reviews* for more information on audit case reviews.

If the findings do not pass the review - e.g. the proposed audit assessment is incorrect - the audit case can go back to the **Audit In Progress** , so that the necessary changes can be made. The audit case can cycle through the **Audit In Progress** and **Review In Progress** states until the findings pass the reviews, in which case, the audit case goes to the **Final Decision** state.

Any related suppressions are released when the audit case is **Closed** .

Audit cases that are in the **Pending** , **Audit In Progress** or **Review In Progress** states can be **Canceled** . Audit case cancellation also cancels any related suppressions.

Refer to the **C1-AuditCase** business object metadata for an illustration of the audit case lifecycle.

Audit Cases Can Cause Suppression

Certain processes or activity may be suppressed while an audit case is ongoing. These processes can include penalty & interest calculation, overdue processing and overpayment processing.

The type of audit case determines whether or not suppression is applicable. Audit case types that cause suppression should specify a suppression type.

Suppressions can be automatically created when the audit case goes into a certain state. The **C1-AuditCase** business object has an algorithm *C1-ADCS-CRSP* that creates and activates suppression when the audit case enters the **Audit In Progress** state.

Changes to the audit start date cause the related suppression's start date to also change.

While an audit is in progress, obligations may get added to or removed from the audit case. If suppression already exists, adding / removing obligations to / from the audit case will also add / remove them to / from the suppressed entities.

Suppressions can be automatically released when the audit case is closed. The **C1-AuditCase** business object has an algorithm *C1-ADCS-RLSP* that does this.

Suppression can be automatically canceled when the audit case is canceled. The **C1-AuditCase** business object has an algorithm *C1-ADCS-CNSP* that does this.

Refer to *The Big Picture of Suppression* for more information about suppression.

Audit Assessments

When the audit identifies necessary changes to the information that the taxpayer originally reported, it usually results in the calculation of a new assessment.

The most common way of determining this new assessment is through an audit form. The form could be the same form type as the one that the taxpayer filed (except that the form is marked as an 'audit' form) or it could be a special audit form type. In either case, the form is brought to a state where the user can review the details of the proposed assessment, see the potential impact to the obligation's balance and make additional changes, as needed.

In some exceptional cases, the adjustment may be created manually (i.e. not via tax form) or certain changes to the taxpayer information are made to trigger a change in the assessed tax (e.g. removing an exemption).

There are two common scenarios:

- The taxpayer being audited did not file a tax return. In this case, the auditor creates the tax form.

- An existing tax return is audited. The auditor creates the audit form by copying the details from the existing form and making the necessary changes. When the audit form posts, the assessment adjustment is created for the difference between the prior assessment and the new assessment.

The **C1-ParentTaxForm** business object provides an Audit action that allows a user to create an audit form from an existing posted form. Any tax forms that inherit from this business object will have the Audit action available. Refer to *Auditing Tax Forms* for more information on auditing existing tax forms.

The audit form is usually not posted until it has gone through proper approval / review and until the taxpayer has accepted the new assessment.

Should the taxpayer disagree with the changed assessment, he/she can file for an appeal. The tax authority may or may not choose to keep the audit open while the appeal is in progress. The audit assessment is not applied to the obligation's balance until a decision is made on the appeal. If the appeal is upheld, the new assessment is not created. On the other hand, if the appeal is denied, the new assessment is posted. Refer to *The Big Picture of Appeals* for more information on appeals.

Audit Case Reviews

The **C1-AuditCase** business object provides for an ability to create reviews from an audit case. This action is available when the audit case's current state supports it.

Refer to *Creating Audit Case Reviews* for more information on how to create audit case reviews.

Reviews have their own lifecycles, which depend on the associated business object. The **C1-AuditCaseReview** business object supplied in base defines a typical lifecycle for an audit case review. It follows a basic lifecycle, as described in *Review Lifecycle*. This business object also has logic to automatically transition the audit case when the last review completes.

Refer to *The Big Picture of Reviews* for more information on reviews.

Assigning Audit Cases To Responsible Users

One or more tax authority users may be assigned as responsible users on an audit case.

A responsible user can be assigned automatically when the audit case is created. The base algorithm *C1-ADCS-ASUR* assigns the user who created the audit case as a responsible user. This algorithm is plugged in on **C1-AuditCase** business object's **Pending** state.

Setting Up Audit Case Options

The following sections describe the objects that must be defined as part of the audit case processing setup process.

Setting Up Audit Case Types

An Audit Case Type defines the configuration information that is common to bankruptcies of a given type. The type of information captured on the audit case type is governed by the audit case type's business object.

To set up an Audit Case Type, select **Admin Menu** > **Audit Case Type**.

The topics in this section describe the base-package zones that appear on the Audit Case Type portal.

Audit Case Type List

The Audit Case Type *List zone* lists every audit case type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent audit case type.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each audit case type.
- Click the **Add** link in the zone's title bar to add a new audit case type.

Audit Case Type

The Audit Case Type zone contains display-only information about an Audit Case Type. This zone appears when an Audit Case Type has been broadcast from the Audit Case Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *C1_AUDIT_CASE_TYPE*.

Setting Up Suppression Types

Audit case types that create suppression must specify a suppression type. Audit-case-related suppression types must specify a suppression entity level of **Obligation** .

Refer to *Suppression: Setting Up Suppression Types* for more details on setting up suppression types.

Setting Up Customer Contact Types For Letters

Define the customer contact types for each type of letter that can be generated from the audit case. Build the extract logic for these customer contact types and define the letter templates accordingly.

Refer to *Setting Up Customer Contact Options* for more details on customer contact types.

Setting Up Review Process Controls For Audit Case Reviews

Audit case types that require reviews must specify a review process control, which defines the types of reviews that are required and the sequence in which they are conducted. The audit case's review process control will default from the value on the audit case type when the audit is created. This default can be overridden.

Refer to *Setting Up Review Options* for details on setting up Review Types and Review Process Controls.

Setting Up Audit Case Reasons

Identify the various reasons for creating an audit case and configure those reasons as Audit Case Reason extendable lookup values.

- Open **Admin Menu** > **Extended Lookup**.
- Search for and select the **Audit Case Reason** extended lookup business object.
- The list of existing audit case reasons are displayed in a standard *List zone*.
- Choose an existing audit case reason to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new audit case reason.

Setting Up Audit Case Sources

If your audit cases can be triggered from a number of different sources, you can configure those sources as Audit Case Source extendable lookup values.

- Open **Admin Menu** > **Extended Lookup**.
- Search for and select the **Audit Case Source** extended lookup business object.

- The list of existing audit case sources are displayed in a standard *List zone*.
- Choose an existing audit case source to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new audit case source.

Setting Up Audit Case Relationship Types

Base values for Audit Case Relationship Types are supplied. You can add your specific relationship types by updating the value for the AUDIT_CASE_REL_TYPE_FLG *lookup* field.

Setting Up Assignment Roles

Base values for responsible user's Assignment Roles are supplied. You can add your specific assignment roles by updating the value for the ASSIGNMENT_ROLE_FLG *lookup* field.

Setting Up An Alert For Highlighting Open Audit Cases

To show a Control Central alert when the account in context has any open audit cases, define an algorithm for the *C1-CCAL-ADCS* base algorithm type and configure the alert in Installation Options.

To configure the alert, open **Admin Menu** > **Installation Options - Framework** and plug in the algorithm you defined in the **Control Central Alert** system event.

Defining Appeal Options

An appeal is an administrative review process used to manage a request from a taxpayer for a change to an official decision by a tax authority. The topics in this section describe how to configure the system to manage appeals.

The Big Picture of Appeals

A taxpayer can lodge an appeal for a variety of reasons. They may wish to dispute certain specific assessments or they may wish to appeal a number of charges related to an assessment such as systemic penalty and interest amounts. If tax returns are adjusted by tax authority staff or new obligations created as a result of an audit case, new assessments may be created that could be disputed.

An appeal process is used to capture the information related to an appeal, track the events involved in reviewing the appeal and record the decision made by each review process.

Taxpayer and Other Related Persons On An Appeal

An appeal process must be linked to a taxpayer. The taxpayer who lodged the appeal can nominate themselves as the primary contact or delegate to an authorized representative such as an accountant. The base package provides the ability to link other persons to the appeal, such as persons who are also financially responsible for the disputed charges or additional representatives such as attorneys. Each related person has a nominated relationship type.

Appeals Are Assigned to Responsible Users

One or more tax authority users may be assigned as responsible users on an appeal.

A responsible user can be assigned automatically when the appeal is created. The base algorithm *C1-AP-ADDUSR* assigns the user who created the appeal as a responsible user with an assignment role of **Primary** .

Appeals Can Cause Suppression

The obligations related to an appeal are usually suppressed from any collection activity or overpayment processing. The system provides the ability to specify which obligations should have activity suppressed as a result of the appeal.

Appeal types that require suppression need to specify a suppression type. The base algorithm *C1-AP-CRTSPR* automatically creates and activates a suppression object for the obligations linked to the appeal when the appeal enters a given state. The **C1-Appeal** business object is configured to create suppression when the appeal enters the **Ready For Review** state.

If an obligation is added to or removed from the appeal, the suppression object is updated accordingly. Refer to base algorithm *C1-AP-PSUCHG* for more details.

The base algorithm *C1-AP-CNLSPR* automatically cancels all suppression objects linked to the appeal when the appeal enters a given state. The **C1-Appeal** business object is configured to create suppression when the appeal enters the **Canceled** state.

The base algorithm *C1-AP-RLSSPR* automatically releases all suppression objects linked to the appeal when the appeal enters a given state. The **C1-Appeal** business object is configured to create suppression when the appeal enters the **Closed** state.

Refer to *Defining Suppression Options* for an overview of Suppression and details on setting up Suppression options.

Related Objects On An Appeal

An appeal can be linked to a number of related objects. Examples include the adjustments or assessments that the taxpayer is contesting, related documents, audit cases or audit forms.

You define appeal related object types by defining the characteristic used to reference the object and linking that characteristic to the appeal related objects entity. For example, to link an adjustment to an appeal you would first define a foreign key characteristic that references the adjustment object. Refer to *Characteristic Types & Values* for more information on setting up characteristic types.

The objects related to an appeal have an associated category. Valid values are defined using the Appeal Related Object category lookup field.

Validating an Appeal

Tax authorities may have rules governing whether an appeal can be accepted for processing. An appeal may be initially assessed to see whether it complies with these rules to determine if it should be rejected without commencing the review or evaluation process. The **C1-Appeal** business object includes a validation state to support this.

It is common for tax authorities to impose time limits on lodging an appeal. The base algorithm *C1-AP-CHKATL* provides an example of logic that validates an appeal's timeliness via a call to an Oracle Policy Automation rule. Refer to *Configuring the Appeal Timeliness Rule* for more details.

If issues are reported during validation, the **C1-Appeal** business object will transition to the **Issues Detected** state. The base algorithm *C1-AP-CRTODO* may be configured on the **Issues Detected** state to send a notification that the appeal is invalid. If the appeal passes validation, the business object enters the **Ready For Review** state.

The **C1-Appeal** business object allows an appeal to be manually transitioned to the **Ready For Review** state from the **Issues Detected** state if a user wishes to override the validation rules. Your implementation may choose to restrict this action to specific user groups using standard application service security configuration.

Canceling an Appeal

The **C1-Appeal** business object provides the ability to cancel an appeal at any point prior to commencing the review process. A monitoring algorithm may be configured to automatically cancel an appeal that has failed validation. The base algorithm *C1-AP-TRNISS* provides an example of logic that transitions an appeal to a canceled state if the appeal has failed due to a specific issue and sufficient time has elapsed to allow for follow up.

Appeals Follow a Defined Review Process

Once an appeal is accepted, it will be reviewed internally by the tax authority. This may involve a conference or hearing with the taxpayer. Once the tax authority has made a decision, the taxpayer may have the option of referring the appeal to other boards of review or judicial courts. The types of reviews available and the order in which the taxpayer can escalate the appeal normally follow a defined sequence.

The system provides the ability to define different types of reviews and a process control that specifies what review types are allowed and in what order for a given appeal process. Appeal types define the default review process control but the user may override this for a particular appeal.

The **C1-Appeal** business object provides a **Create Review** action that allows the user to add a new review sub-process. Users are presented with a list of the next allowable review steps, based on the reviews already undertaken for the appeal. Only one review can be active for an appeal at a time, so the action is only available if the most recent review linked to the appeal is closed.

An appeal moves from the **Ready For Review** state to **Review In Progress** when the first review is created. The appeal remains in that state, allowing the user to continue to add successive review events until the taxpayer accepts the response to the latest review or the appeal has passed through all possible internal and external review types.

Refer to *Defining Review Options* for more information on configuring reviews.

Appeal Reviews

The lifecycle of a review depends upon the configuration of the associated business object. The base **C1-AppealReview** business object defines a typical review for an appeal process. It follows a basic lifecycle, as described in *Review Lifecycle*.

In addition to the common functionality, a base algorithm is provided to create a customer contact for an appeal based on the response type. The base algorithm type *C1-RVW-CCCRR* is designed to be configured on the appeal **Response Recorded** state.

Closing Appeals

Once the last review process is complete, the appeal can be manually transitioned to the **Final Decision** state. The base algorithm *C1-AP-CINCRE* provides the ability to prevent an appeal from being transitioned to **Final Decision** before the most recent review is complete.

The appeal remains in the **Final Decision** state while users make the appropriate corrections to the taxpayer's account, depending on the outcome of the appeal. When no further action is required, the appeal can be transitioned to **Closed** .

The base algorithm *C1-AP-CPTODO* can be configured on the **Closed** or **Canceled** states to complete any outstanding to do entries for the appeal .

Monitoring Appeals

The actions required to progress an appeal are mostly manual. An appeal may be waiting for a review to be scheduled. Appeals can remain open while the taxpayer has the ability to escalate the appeal in response to the most recent decision. Periodic monitoring provides the ability to check whether an appeal has not progressed as expected.

The base algorithm *C1-AP-APWL* provides the ability to send a notification that the appeal has been waiting in a given state for too long. The algorithm is designed to be configured on the **Ready For Review** and **Final Decision** states.

The base algorithm *C1-RVW-CCCRR* provides the ability to send a notification that the appeal has been waiting in the **Review In Progress** state for too long after the latest review was completed.

Setting Up Appeal Options

The following sections describe the objects that must be defined as part of the appeal processing setup process.

Setting Up Appeal Types

Appeal Types contain the rules that control how appeals are processed.

To set up an appeal type, select **Admin Menu** > **Appeal Type**.

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

Appeal Type List

The Appeal Type *List zone* lists every appeal type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent appeal type.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each appeal type.
- Click the **Add** link in the zone's title bar to add a new appeal type.

Appeal Type

The Appeal Type zone contains display-only information about an appeal type. This zone appears when an appeal type has been broadcast from the Appeal Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *C1_APPEAL_TYPE*.

Setting Up Review Process Controls

An appeal type must be linked to a review process control, which defines the types of reviews required to process the appeal and the sequence in which they are conducted. The appeal's review process control will default to this value when an appeal of this type is added but may be overridden.

Refer to *Setting Up Review Options* for details on setting up Review Types and Review Process Controls.

Setting Up Suppression Types

An appeal type may be optionally linked to a suppression type, which defines the suppression entity level and the processes to be suppressed. If an appeal type references a suppression type, a suppression object of that type is automatically created when the appeal enters the appropriate state. Appeals suppress activity on selected obligations, so appeal types should reference suppression types with an entity level of **Obligation** .

Refer to *Setting Up Suppression Options* for details on setting up Suppression Types.

Setting Up Customer Contact Types

The base Review Response Type extendable lookup business object allows users to specify an optional customer contact type and class to be used when notifying taxpayers of the response given to their appeal review.

To set up a customer contact type, select **Admin Menu** , **Customer Contact Type** . Refer to existing help on setting up customer contact types.

Setting Up an Alert to Highlight Open Appeals

The base algorithm type *C1-AP-OPEN* provides the ability to create a Control Central Alert if open appeals exist for the person or account in context.

To configure the alert, navigate to **Admin** > **Menu, Installation Options - Framework** and configure the base algorithm using system event **Control Central Alert** .

Configuring the Appeal Timeliness Rule

In order to use the base algorithm type that invokes the Oracle Policy Automation rulebase to validate an appeal's timeliness, you must define additional components, as follows:

- Define a web service adapter for the rule. Refer to *OPA configuration* for more details.
- Define a message category and number to be used to indicate that the appeal has failed the rule in the appeal's issues list. To set up a message, navigate to **Admin** > **Menu, Message** and add a new message for the 'Implementer's Messages' category.
- Create an algorithm for algorithm type *C1-AP-CHKATL* that references the web service adapter, message category and number as input parameters.
- Configure the algorithm on the **Validated** state for the appropriate appeal business objects, using the **Enter** system event.

Setting Up Appeal Relationship Types

Base values for Appeal Relationship Types are supplied. You can add your specific relationship types by updating the value for the APPEAL_REL_TYPE_FLG lookup field.

Refer to *Defining Extendable Lookups* and *Extendable Lookups - Main* for more information on lookups.

Setting Up Assignment Roles

Base values for a responsible user's Assignment Role are supplied. You can add your specific assignment roles by updating the value for the ASSIGNMENT_ROLE_FLG lookup field.

Refer to *Defining Extendable Lookups* and *Extendable Lookups - Main* for more information on lookups.

Setting Up Appeal Related Object Categories

You can add your specific appeal related object categories by updating the value for the APPEAL_REL_OBJ_CAT_FLG lookup field.

Refer to *Defining Extendable Lookups* and *Extendable Lookups - Main* for more information on lookups.

Defining Review Options

Some common case management processes such as appeals and audits, include steps in which the case is evaluated either internally by different groups within the tax authority, or externally by other agencies, such as judicial courts. These review processes typically follow their own lifecycle and may be part of a sequence of reviews that are escalated to higher levels of authority. The topics in this section describe how to configure the system to manage reviews.

The Big Picture of Reviews

A review provides the functionality to record key dates and determinations of a review process. Reviewers may be required to make a decision by a certain date and taxpayers may be given a fixed period of time to respond. A predefined list of responses is used to ensure review decisions are described and classified consistently. External reviews may be linked to the person or entity conducting the review, such as a court.

Defining Allowable Review Types and Sequences

A case management process may trigger more than one review. The types of reviews that can occur differ according to the process type. In addition, the order in which the allowable review types are conducted often follows a defined sequence. For instance, if a taxpayer lodges an appeal with a tax authority and disagrees with their decision, they may have the right to refer the case to one or more external courts. Audit cases may need to be approved by more than one supervising group before any action is taken.

Review process controls are used to define a particular sequence or hierarchy of review types. Users can specify what review types are allowed and in what order for a specific process control. The review process control also allows the user to indicate whether a given review type may be bypassed or repeated.

Creating Reviews

A review is created as a result of a user-initiated action invoked from a case management process. The base package supports the creation of reviews from an appeal or audit case. The **Create Review** action in the appeal or audit case portal zone guides the user to select the next available review type as governed by the associated review process control. The applicable review process control is defined on the appeal or audit case type.

It is important to note that a review exists as a sub-process linked to another object and cannot be added or viewed independently of the primary process. Refer to the documentation on *Reviewing an Appeal* and *Creating Audit Case Reviews* for more details on how to create and maintain reviews.

Review Lifecycle

The lifecycle of the review depends upon the configuration of the associated business object. The base package includes review business objects that are specific to appeals and audit cases that have the same simple lifecycle, as follows:

- The review is initially created in the **Pending** state.
- A **Pending** review may be manually transitioned to **Canceled**.
- The user can manually transition the review to the **In Progress** state once key information has been recorded or dates have been set. A review that has transitioned to **In Progress** cannot be canceled—it must progress through the **Response Recorded** and **Closed** states. A monitoring algorithm can be configured to check whether the review has been waiting too long in the pending or in progress states. Base algorithm *C1-RVW-TDRPD* provides an example of this logic.
- The user can manually transition the review to the **Response Recorded** state when a decision has been reached and the review has been updated with the appropriate review response code. A monitoring algorithm can be configured to automatically transition the review to the **Response Recorded** state when the review decision date has been populated and is on or before the business date. Base algorithm *C1-RVW-TRTRR* provides an example of this logic.
- The user can manually transition the review to the **Closed** state once all activities are complete. An algorithm can be configured to complete all open to do entries that are linked to the review. Base algorithm *C1-RVW-CMPTD* provides an example of this logic.

Setting Up Review Options

The following sections describe the objects that must be defined as part of the review processing setup process.

Setting Up Review Types

Review Types contain the rules that control how reviews are processed.

To set up a Review Type, select **Admin Menu** > **Review Type**.

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

Review Type List

The Review Type *List zone* lists every review type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent review type.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each review type.
- Click the **Add** link in the zone's title bar to add a new review type.

Review Type

The Review Type zone contains display-only information about a Review Type. This zone appears when a Review Type has been broadcast from the Review Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *C1_REVIEW_TYPE* .

Setting Up Review Process Controls

Review types are logically grouped together in a review process control. The sequence of the review types in the review process control governs the order in which reviews may be created for processes associated with that control.

To set up a Review Process Control, select **Admin Menu** > **Review Process Control** .

The topics in this section describe the base-package zones that appear on the Review Process Control portal.

Review Process Control List

The Review Process Control *List zone* lists every review process control. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent review process control.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each review process control.
- Click the **Add** link in the zone's title bar to add a new review process control.

Review Process Control

The Review Process Control zone contains display-only information about the review process control. This zone appears when a review process control has been broadcast from the Review Process Control List zone or if this portal is opened via a drill down from another page.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *C1_REVIEW_PROC_CTRL*.

Setting Up Review Response Types

Review response types are defined using an extendable lookup. To view or create review response types:

- Open **Admin Menu** > **Extendable Lookup**
- Search for and select the **Review Response Type** extendable lookup business object.
- The list of existing review response types is displayed in a standard *List zone*.
- Choose an existing review response type to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new review response type.

The base Review Response Type extendable lookup business object allows users to specify an optional customer contact type and class. Refer to *Appeal Reviews* for details on how customer contacts are used in appeal reviews.

Defining Suppression Options

Suppression is a common function to stop certain events from taking place on a taxpayer's account. Common causes of suppression are processes such as bankruptcy cases, appeals or other internal investigations such as audit cases. The topics in this section describe how to configure the system to manage suppression.

The Big Picture of Suppressions

A tax authority may want to suppress activity on a taxpayer's account for a number of different reasons. In some cases it is a matter of internal policy, where the tax authority suppresses certain activities while processing an appeal or audit case. In other instances, it may be a statutory requirement imposed by the regulations surrounding a process initiated externally, such as a bankruptcy proceeding.

Some common types of activities that may be suppressed are penalty and interest calculation, overdue processing and overpayment processing. Levels of suppression can be large or small, encompassing a specific account, a tax role or a single obligation. Suppressions may be initiated and released manually or created and released automatically by other processes.

A suppression object is used to capture the specific entities and processes that are subject to a given suppression event.

Suppression Types

Suppression types define the rules governing how suppressions are created and managed. Certain types of suppressions may be created manually while others should only be created automatically by other processes. The processes affected by a suppression event are defined on the suppression type. Users may choose to add to or remove suppressed processes when the suppression is added manually.

Some types of suppression may be configured to be automatically released after a given period of time whereas other must be released manually.

Suppressing Process Activity

The activities that need to be suppressed can vary according to the reason for suppression - for example, if a tax authority receives notice that a bankruptcy proceeding is in place that affects taxes owed, penalty and interest (P&I) calculation and overdue processing activity must be suppressed until the bankruptcy case is closed.

In most cases, processing is put on hold starting at the suppression's start date and resumes on the suppression's release date. Penalty and interest calculations are an exception to this, as P&I may be recalculated to accommodate retroactive changes to the obligation's balance. Refer to *How Suppression Affects Penalty and Interest* for more details on P&I specific processing.

A suppression event does not actively "hold" work but rather affects how the associated processes work . It is the responsibility of the associated processes to consider effective suppressions whenever actions are initiated. The way in which suppression affects processing can differ according to the activity being suppressed. For example:

- P&I provides the ability to turn off P&I rules for the suppressed period
- Overdue Processing has a plug-in designed to hold event activation
- Business object -based processes such as overpayment processing need algorithms that suppress actions like state transitions

The system provides sample algorithms that can be used to suppress some common activities. Refer to *Configuring Overdue Processes For Suppression*, *Configuring Overpayment Processes For Suppression*, and *Configuring Obligation Types For P&I Suppression* for a description of how these processes can be suppressed.

Suppressed Entity Level Defines Suppression Scope

Suppression can apply at one of four entity levels - person, account, tax role or obligation. The base package provides common logic to determine if suppression is in effect for an entity. This logic considers all suppression records for the entity and its higher level entities.

Suppression scope can be summarized as follows:

- A person-level suppression object will also cause suppression on all the accounts for which that person has financially responsibility, all tax roles for those accounts and all obligations for those accounts.
- An account-level suppression object will also cause suppression on all tax roles for that account and all obligations for that account.
- A tax role-level suppression object will also cause suppression on all obligations for that tax role.
- An obligation-level suppression object will cause suppression only on that obligation.

It is possible for a given entity, such as an obligation, to be affected by suppressions with differing periods or at differing levels. For example, a suppression record may be manually added for an account with one or more obligations. A subsequent bankruptcy case may be created that includes one of those account's obligations. The system assumes that both suppressions are in effect for the obligation and that the period of suppression starts from the start of the earlier suppression record (for the account) until the later of the two release dates.

Note that the release date of a suppression event is the date on which the suppression ceases to have effect. Processes that are put on hold due to suppression will restart processing on the release date - for example, overdue events will

be triggered on the day of release. For P&I calculations, P&I will be suppressed from the start date of the suppression until the day prior to the release date, inclusive.

Refer to the base business service **C1-GetEffectiveSuppressions** for more details on the common logic used to retrieve suppression details and periods.

How Suppression Affects Penalty and Interest

P&I processing caters for retroactive changes to an obligation's balance. It needs to consider suppressions that were in effect during the recalculation period even though the suppressions may now be released . The logic that gets effective suppressions considers both Active and Released states for P&I processing - these states are marked as 'P&I In Effect'. The result is that P&I rules are permanently turned off for the period of the suppression event.

If a suppression event is cancelled, it is regarded as if it had been deleted and no longer has an effect. Penalty and interest calculations are updated to reverse any prior effect of the suppression.

The base package provides a **Business Object - Post-processing** algorithm and a **Business Object Status - Enter** algorithm for the **C1-Suppression** business object that calculates P&I for entities affected by the suppression whenever key events occur. Refer to base algorithms *C1-SUPP-PI* and *C1-SUPP-UPI* for more details.

Your implementation may wish to prevent certain adjustments or charges being made to a taxpayer's obligations will suppression is in effect. The system provides an Adjustment Type Validation algorithm designed to return an error if suppression applied to the adjustment's obligation. Refer to *Configuring Adjustment Types For P&I Suppression* for more details.

Other Processes Create Suppression

Many suppression events will be created automatically as the result of initiating another process, such as a bankruptcy case or an appeal.

The process that causes the suppression is responsible for creating the suppression object with its associated entities and processes. The initiating process is also responsible for ongoing maintenance of the suppression object, including:

- Updating the related suppression date when a key date on the process is changed, such as an appeal submission date
- Adding or removing suppressed entities if the related entities are removed or added to the initiating process
- Cancelling or releasing the suppression at the appropriate time

Refer to *The Big Picture Of Bankruptcy*, *The Big Picture Of Appeals*, and *The Big Picture Of Audit Cases* for examples of processes that create and manage suppressions.

Auditing Suppressions

Because suppression events can cause retroactive changes to penalty and interest, key changes are audited. The base package provides **Business Object - Audit** algorithms for the **C1-Suppression** business object that log changes to the suppression dates and suppressed entities. Refer to base algorithms *C1-SUPP-DTCH* and *C1-SUPP-EDEL* for more details.

Suppression Lifecycle

The lifecycle of a suppression object depends upon the configuration of the associated business object. The base package includes a sample **C1-Suppression** business object with a simple lifecycle, as follows:

- The suppression is initially created in the **Pending** state. The suppression has no effect while in this state - the details may be edited without triggering other processing such as P&I updates. A deferred monitoring algorithm can be configured to automatically transition the suppression to **Active** if the start date is on or before the system date. Base algorithm *C1-SUPP-AC* provides an example of this logic.

- The user can manually transition the suppression to the **Active** state once the details are finalized. Suppressions that are created by other processes will typically be activated by the initiating process. An enter algorithm may be configured to trigger validation specific to the **Active** state. The base algorithm *C1-SUPP-VAAC* provides an example of this logic.
- A monitor algorithm may be configured on the **Active** state to check for suppressions that have been open too long. The base algorithm *C1-SUPP-MAXD* provides an example of this logic. An exit algorithm can be configured to complete all open monitoring to do entries when the suppression is no longer active. Base algorithm *C1-SUPP-CTDO* provides an example of this logic.
- A suppression object can be released by transitioning to the **Released** state or by updating the end date and having the suppression transition automatically. Suppressions that are created by other processes will typically be released by the initiating process. A monitor algorithm may be configured on the **Active** state to check for suppressions whose end date is populated and automatically transition them to **Released** . The base algorithm *C1-SUPP-REL* provides an example of this logic. An enter algorithm may be configured to provide a default end date or validate the existing end date for a suppression transitioned to **Released** . The base algorithm *C1-SUPP-END* provides an example of this logic.
- A suppression object can be **Canceled** from the **Pending** or **Active** states. Suppressions that are created by other processes will typically be canceled by the initiating process if appropriate.
- If the suppression affects penalty and interest, P&I should be updated for the related obligations when the suppression period starts and ends. An enter algorithm may be configured to trigger P&I updates when the suppression enters the **Active** , **Released** or **Canceled** states. Base algorithm *C1-SUPP-UPI* provides an example of this logic

Setting Up Suppression Options

The following topics describe the objects that must be defined as part of suppression processing setup.

Setting Up Suppression Types

Suppression Types contain the rules that control how suppressions are processed.

To set up a Suppression Type, select **Admin Menu** > **Suppression Type**.

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

Suppression Type List

The Suppression Type *List zone* lists every suppression type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent suppression type.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each suppression type.
- Click the **Add** link in the zone's title bar to add a new suppression type.

Suppression Type

The Suppression Type zone contains display-only information about a Suppression Type. This zone appears when a Suppression Type has been broadcast from the Suppression Type List zone or if this portal is opened via a drill down from another page.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *C1_SUPPRESSION_TYPE*.

Configuring Overdue Processes for Suppression

The base package includes a sample algorithm that provides an example of logic that determines if overdue event activation should be placed on hold when suppression applies. It is designed to be used for overdue processes that collection on obligations or assessments.

In order to use the base algorithm type to suppress overdue events, you must perform the following configuration:

- Create an algorithm for algorithm type *C1-OVRD-SUPP* . Refer to the algorithm type description of an explanation of the parameters required for the algorithm and their use.
- Navigate to **Admin Menu** > **, Overdue Process Template** and configure the algorithm using the **Hold Event Activation Criteria** system event for each of the overdue process templates to which suppression applies.

Refer to *Setting Up Overdue Process Templates* for more information.

Configuring Overpayment Processes for Suppression

Business object based processes should be configured to check for suppression before triggering processing that should be placed on hold while suppression applies.

The base package includes sample algorithms for the Overpayment Process maintenance object that demonstrate how this can be achieved.

The base algorithm type *C1-OP-SUPPCH* provides an example of an enter algorithm that will issue an error if suppression applies to the overpayment process's obligation. This algorithm type would typically be configured on any state for the overpayment process business object that would affect the associated obligation's financial balance. Its purpose is to prevent online transitions to the invalid states.

The base algorithm type *C1-OP-SUPPMN* provides an example of a monitor algorithm that will indicate that transition should not take place if suppression applies to the overpayment process's obligation. This algorithm type would typically be configured on any state that is set up to automatically transition to a state for the overpayment process business object that would affect the associated obligation's financial balance. Its purpose is to prevent background transitions to the invalid states.

Refer to *Business Objects* for more information on business object lifecycles and configuration.

Configuring Obligation Types for P&I Suppression

The base functionality that supports penalty and interest calculations is designed to recognize discrete time periods in which differing calculation rules apply. This includes identifying periods where penalty and interest should not accrue. The base package supports functionality to suppress P&I calculation for a given period by providing a sample algorithm that identifies all the suppressed periods that apply to the overall calculation period and turns off all P&I rule processing for those date ranges.

In order to use the base functionality that suppresses P&I, you should configure algorithm *C1-PI-PR-SUP* on all appropriate obligation types. Refer to the algorithm type description for a detailed description of the way in which the periods of suppression are determined.

Refer to *The Big Picture Of Penalty and Interest* for more information on configuring penalty and interest rules.

Configuring Adjustment Types for P&I Suppression

In order to use the base algorithm type that validates whether a particular adjustment type can be created when suppression applies, you must perform the following configuration:

- Create an algorithm for algorithm type *C1-RVW-CCCRR* or use the base algorithm that applies the validation rule to all suppressed process types.
- If you wish to restrict the validation to suppressions of particular process types, create multiple algorithms specifying those process types as their input parameter.
- Navigate to **Admin Menu** > **Adjustment Type** and configure the appropriate algorithm using the **Validate Adjustment** system event for each of the adjustment types that should not be created. Multiple **Validate Adjustment** algorithms may be configured to cover multiple processes for the same adjustment type.
- Refer to *Adjustment Type - Algorithms* for more information.

Setting Up Suppressed Process Types

The types of processes that can be affected by suppression are defined using an extendable lookup. To view or create suppressed process types:

- Open **Admin Menu** > **Extended Lookup**.
- Search for and select the **Suppressed Process Type** extendable lookup business object.
- The list of existing suppressed process types is displayed in a standard *List zone* zone.
- Choose an existing suppressed process type to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new suppressed process type.

Setting Up Suppression Creation Reasons

Suppression creation reasons are defined using an extendable lookup. To view or create suppression creation reasons:

- Open **Admin Menu** > **Extended Lookup**.
- Search for and select the **Suppression Creation Reason** extendable lookup business object.
- The list of existing suppression creation reasons is displayed in a standard *List zone* zone.
- Choose an existing suppression creation reason to view, edit, delete or duplicate.
- Use the **Add** link in the zone header to create a new suppression creation reason.

Alert to Highlight Active Suppressions

The base algorithm type *C1-CCAL-ASE* provides the ability to create a Control Central Alert if active suppressions exist for the person or account in context.

To configure the alert, navigate to **Admin Menu** > **Installation Options - Framework** and configure the base algorithm using system event **Control Central Alert** .

Defining Pay Plan Options

Tax authorities use pay plans as a collections tool. When a taxpayer agrees to be on a payment plan, it is an acknowledgement of his/her tax debt. Pay plans are typically used to satisfy obligations in which the taxpayer cannot satisfy the total obligation with one lump payment.

The pay plan's terms are typically negotiated between the tax authority and taxpayer. The payment amount and specific date intervals of the pay plan include such factors as total obligation amount and the taxpayer's ability to pay. Payments received are used to satisfy tax, penalty, and interest.

The Big Picture of Pay Plans

A pay plan is a special type of payment plan that encompasses two major elements:

- A set of scheduled payments
- The business rules used to recommend the payment schedule

Pay plans are managed via an obligation whose obligation type has a special role of **Pay Plan**. This type of obligations must have one or more recommendation rules, and are allowed to have payment schedules.

A pay plan's status is just like any other obligation's status. An active pay plan implicitly has a "kept" status (i.e. all scheduled payments have been made). A stopped plan implies that the pay plan has been completed, i.e. paid off, or stopped due to non-payment. It is important to note that the scheduled payments have their own status.

A list of the obligations covered by a pay plan is maintained with the pay plan. This list is used at payment distribution to determine which obligation to pay. An obligation may be covered by a pay plan when its obligation type is flagged as **Eligible for Pay Plan**.

Creating Pay Plans

A payment plan can be created as a result of the following scenarios:

- A collections case is created as a result of overdue debt. One of the available actions for the user responsible for the collections case is setting up a pay plan. Refer to *The Big Picture of Collection Cases* for details on how a collection case can create a pay plan.
- The taxpayer may also voluntarily seek to enter a payment plan. Refer to *Maintaining Pay Plans* for details on how to create a pay plan manually.

Activating Pay Plans

A pay plan is created in a **pending start** status. For the pay plan to become effective, it needs to be activated. If the pay plan does not require any special approval, the user can activate the pay plan once they have completed setting up the payment schedule. This can be done by navigating to the obligation page, and using the **activate** button.

If the pay plan requires approval, a different process might be implemented depending on the tax authority's rules.

When a pay plan is activated, you can perform special processing using an algorithm plugged in on the obligation Activation system event on the pay plan obligation 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 taxpayer of their payment amount and payment schedule.
- Initiate the creation of a payment coupon book for a taxpayer.
- In some cases, the taxpayer may be charged a setup fee for the pay plan. See the base algorithm (*C1-OT-LEVFEE*) for an example.

The system comes supplied with a sample algorithm type (*SAAT-CC*) that simply creates a customer contact to indicate that the pay plan is activated.

Breaking Pay Plans

If the taxpayer does not meet their terms of the pay plan, the pay plan is considered broken. A pay plan can be broken in the one of the following ways:

- The user has used the **Break** action on the pay plan page.
- An automated process, such as monitor algorithm has determined that the taxpayer has broken the terms of the pay plan.

When the pay plan is broken, the pay plan is transitioned to **pending stop** status.

A base package break pay plan algorithm type (*C1-PAYP-BRK*) can be used to create a characteristic value to indicate if a pay plan is manually stopped by a user.

In addition, the algorithm type (*C1-OT-TPPRVW*) will create an account monitor review trigger if the pay plan is not associated with a collections case.

You can plug in an algorithm (of type *SAIS-ST*) on the obligation Stop Initiation system event on the pay plan obligation type to automatically stop the obligation (i.e. transition it to **stopped** status).

To finalize a pending stop obligation, the system calls the stop obligation algorithm plugged-in on the obligation Stop system event on the obligation type.

Stopping Pay Plans

If the taxpayer pays off the outstanding debt, the pay plan is considered stopped. A pay plan can be stopped in one of the following ways:

- The collection case that created the pay plan was closed as a result of the taxpayer paying off the debt, and updates the pay plan to **pending stop**.
- An automated process, such as monitor algorithm has determined that the taxpayer has paid off the debt, and updates the pay plan status to **pending stop**.

As indicated above, an algorithm can be plugged it to transition the pay plan immediately from **pending stop** to **stopped**.

Canceling Pay Plans

A pay plan may need to be cancelled under certain circumstances that are initiated either by the tax authority or the taxpayer. For example:

- A user may need to cancel an erroneous pay plan after it has already been activated.
- A tax authority may cancel pay plans (en masse) in the event of natural disasters.
- When a taxpayer indicates additional hardship, the procedure may involve canceling the existing pay plan and renegotiating a new pay plan.

Canceling a pay plan causes the pay plan status to change to **Cancelled**. Any specific actions that need to take place can be plugged in on obligation Cancel system event. The base package algorithm type (*C1-OT-CRCC*) can be used to create a customer contact when the pay plan is cancelled. Alternatively, (*SACA-CRTODO*) can be used to create a to do entry when the pay plan is cancelled.

Distributing Payments for Pay Plans

For pay plans, payments are distributed directly to the covered obligations using payment distribution rules. A tax authority's business processes will dictate the distribution of payments to the corresponding obligations. For example:

- A tax authority wants payments to be applied to the oldest obligation first. It them moves to the next oldest obligation until all obligations are satisfied. The oldest obligation is given highest priority.
- A tax authority wants payments to be applied to obligations attached to an active collections case. The obligations linked to a collections case are assigned highest priority.
- The base-package provides an algorithm that will distribute payments to obligations covered by a pay plan based on age of debt, and priority indicated on the obligation type (*C1-DR-PAYPP*).

Refer to *Payment Event Distribution Rules* for details.

Processing Scheduled Payments

In addition to the pay plan having a status, the scheduled payments each have their own status. A scheduled payment is created in a **pending** status when the pay plan is activated.

Each scheduled payment is reviewed by the base base-package batch **C1-PAYPS** on the payment due date. The batch will set the scheduled payment status to **processed** after executing the **Process Pay Plan Scheduled Payment** plug-in defined on the obligation type.

This plug-in can contain specific business logic that need to be executed when a scheduled payment is due. Based off the rules that will be configured by the tax authority, a multitude of actions can occur. They range from giving the taxpayer a few extra grace days to make the payment to canceling the current payment plan and forwarding the obligations to a collections authority. A tax authority's business rules as well as each taxpayer's personal history and circumstances should be taken into account when determining what actions to take next.

The base-package provides an algorithm (*C1-CC-CHKSPP*) that will create a to do if a scheduled payment has not been paid. It will also detect when actual payments have fulfilled the pay plan in advance.

If your implementation's rules require additional status values (e.g. kept, late, partial), you will need to do the following:

- Create your own copy of **C1-PAYPS**.
- Add custom values to lookup **NB_SCHED_PAY_STATUS_FLG**.

Automatic Payment and Pay Plans

If a taxpayer wants to pay their pay plan scheduled payments automatically, the account must be set up for automatic payment. In addition, the pay plan must indicate that automatic payment is being used.

Fastpath:

Refer to *How To Set Up Automatic Payment For A Pay Plan* for more information.

When this is done, a background process referred to as **C1-PAYPA** creates automatic payments on the scheduled payment date by calling the automatic payment creation algorithm plugged in on the installation record.

The base-package provides two auto pay creation algorithms that support pay plans scheduled payments:

- Use algorithm type *APAY-CREATE* if your implementation distributes payments using the standard account type payment distribution.
- Use algorithm type *C1-APAY-CRDR* if your implementation distributes payments using distribution rules.

Alerts For Pay Plans

The system provides *alerts* to highlight the existence of pay plans. These alerts are important to assist the tax authority's users:

- An alert is displayed if the account has a pay plan that is not stopped (e.g. **pending start**, **active** or **pending stop**).
- For taxpayers who are permanently forbidden from having a pay plan, the user should put a permanent *alert on the account*.
- Use an algorithm to highlight cancelled pay plans with an entry in the alert zone. The algorithm type to do this is not provided. Use *C1-STOP-SA*as an example of how to create this type of algorithm.
- Configuring a specific *customer contact type* with the alert algorithm type (*CC BY TYPCL*) can cause alerts to be displayed whenever a user adds a *customer contact* of this type. This method can be used to highlight special conditions relating to the pay plan.

Fastpath:

For more information about introducing alert conditions on Control Central, refer to *Installation Options - Algorithms*.

Setting Up Pay Plan Options

The topics in this section describe functionality that you must consider when designing pay plans.

Designing Recommendation Rules

This section describes topics related to designing your recommendation rules.

What is a Pay Plan Recommendation Rule

Users ask the system to recommend the scheduled payments for a pay plan. In general, this recommendation process must establish:

- The amount to be paid
- The dates on which the payments are due

A recommendation rule comprises of three elements:

- Payment schedule algorithms
- An algorithm to calculate an average daily amount. This is optional. If this plug-in must be used, it should be invoked from your payment schedule algorithms
- A collection of default parameter values for the payment schedule algorithm type

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 pay plans already in effect. Existing pay plans keep the parameter values that were used when the pay plan 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).

Note:

Normally parameter values for an algorithm type are kept with the algorithm. Because the parameters may vary for each pay plan, while the same algorithm logic is used, the parameter values are kept with the pay plan.

Examples of Recommendation Rules

The following are two common methods of calculating a schedule of payments:

- Taxpayer knows how much and when they can pay. The recommendation rule will calculate the schedule of payment dates given the fixed payment amount, frequency, and the first payment date. For example: monthly payments of $1,000 beginning on August 1, 2007.
- Taxpayer knows when they would like to pay the debt by, and how often they can pay. The recommendation rule will calculate the schedule of payment dates and payment amounts given the first payment date, the last payment date and the frequency. For example: monthly payments to be made starting on August 1, 2007 and ending on February 1, 2008.

Generate Payment Schedule Algorithm Types

- The core of the pay plan recommendation rule is the Generate Payment Schedule plug-in. The base-package provides two methods for generating a payment schedule: a fixed amount schedule or a fixed duration schedule. Both methods incorporate forecasted penalty and interest charges into the calculations of scheduled payment amount. For more details on these plug-ins see the following:
- A fixed amount payment schedule *C1-PP-FIXAMT*.
- A fixed duration payment schedule *C1-PP-FIXDUR*.

Penalty and Interest Considerations

If *penalty and interest* (P&I) rules are applicable to the obligations covered by a pay plan, the payment schedule that is generated by the recommendation rule's generate payment schedule algorithm should factor expected penalty and interest calculations.

The penalty and interest plug-in spot may be called in *forecast* mode to forecast the charges to a certain date in the future. The plug-in spot also supports receiving financial transactions supplied by a calling service allowing for

"what if" scenarios to be supplied. For example, when forecasting P&I to the future, the generate payment schedule algorithm can include proposed scheduled payments along with existing financial transaction to the P&I algorithm to get a more accurate estimate of the future P&I charges.

Setting Up The System To Enable Pay Plans

The above topics provided background information about how pay plans are supported in the system. The topics in this section describe how to set up the system to enable pay plan functionality.

Characteristic Type

If your implementation requires a characteristic added to the pay plan when the pay plan is broken, you will need to create a characteristic type that specifies **obligation** as its characteristic entity.

Customer Contact Class And Types

If your implementation requires certain customer contacts and/or letters created when the pay plan is activated or stopped, you will need to create appropriate customer contact class and customer contact type(s).

Note:

If you want to send letters to your taxpayers when a contact of any of these types is created, you must create an appropriate *letter template* and attach it to the customer contact type.

Algorithms

As described above, in *The Big Picture of Pay Plans*, the base package provides a number of algorithm types for different system events in the pay plan's lifecycle. In order to use any one of these algorithm types, you will need to create an associated algorithm and ensure any required parameters are populated.

If your implementation created additional custom algorithm types, you will also need to define associated algorithms.

Defining a Pay Plan Recommendation Rule

Recommendation rules are used to recommend scheduled payments for pay plans. To define recommendation rules, navigate to **Admin Menu, Pay Plan Recommendation Rule.**

Description of Page

Enter an easily recognizable **Recommendation Rule** code and **Description** for each recommendation rule.

If applicable, 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 pay plans that use this recommendation rule. The Payment Schedule Algorithm Type cannot be modified if a pay plan 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.
- **Pay Plan 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*.

Obligation Types

This section provides information about setting up the obligation types for the pay plan itself and the obligation types for covered obligations.

Obligation Type for Obligations Covered by Pay Plans

Consider which types of obligations need to be covered by a pay plan. For each obligation type that should be allowed to be covered by a pay plan, ensure that the **Eligible for Pay Plan** flag (on the Billing page) to **Eligible**.

Pay Plan Obligation Type

In order to enable pay plan functionality, you will need to define an obligation type that is suitable for a pay plan. Depending on the business rules of the tax authority, you may need multiple pay plan types, which will require you to define multiple obligation types. For example you might want to create a pay plan type for individual taxpayers, and a different pay plan type for a business.

The following points provide guidelines for creating a pay plan obligation type:

Obligation Type - Main

- **Tax Type** should either reference a specific tax type or a generic tax type (e.g. miscellaneous fees) depending on your authority's business rules.
- **Revenue Class** should be set to **N/A**. (Revenue classes are not applicable because pay plans do not apply a rate and revenue classes are only relevant for obligation types that use a rate.)
- The **Payment Segment Type** should reference the **Normal Payment**.
- **Do Not Overpay** should be on. Payments are not distributed directly against the pay plan.

Obligation Type - Detail

- **Special Role** is **Pay Plan .**

Obligation Type - Billing

Eligible for Billing flag should be off, as pay plans do not get billed.

Additionally, the **Characteristic Location Required** should not be checked for pay plans.

Eligible for Pay Plan should be set to **Ineligible**. This option only applies to the obligation types that can be covered by a pay plan.

Obligation Type - Rate

Pay plans do not use rates, so the **Rate Required** flag should be off.

Obligation Type - Algorithms

Plug in any algorithms defined above for the following system events:

- Break Pay Plan
- Initiate Stop
- Obligation Activation
- Obligation Cancel

- Obligation Stop
- Process Pay Plan Scheduled Payment

Obligation Type - Pay Plan Recommendation Rule

Add the recommendation rules defined above that are valid for obligations of this type. Also, indicate which recommendation rule should be used as the default.

Pay Plan Background Processes

Ensure that the following background processes are scheduled:

- Pay Plan Scheduled Payment Processing (**C1-PAYPS**)
- Pay Plan Scheduled Payment Automatic Payment Create (**C1-PAYPA**)

Defining Work Management Options

This section describes functionality provided by the product to manage business processes in situations where the base product doesn't include a dedicated maintenance object. The base product recommends using the process flow for this purpose. The case functionality is legacy functionality.

Configuring Process Flows

Process Flow is a business object based maintenance object provided to support workflow-type business processes or tasks when existing base product maintenance objects are not well suited for the task. The topics in this section describe the generic Process Flow entity and how it can be customized.

The Big Picture of Process Flows

The following topics describe process flow functionality.

Process Flow Is A Generic Entity

The Process Flow maintenance object is a generic entity that can be configured to represent custom entities and support automated workflows for a variety of applications. Each process flow references a business object to describe the type of entity it is. The base package does not provide any pre-configured business objects for Process Flow. Implementations configure their own business objects and related user interface to orchestrate the desired custom business process.

A status column on the process flow may be used to capture its current state in the processing lifecycle controlled by its business object.

The maintenance object also supports a standard characteristic collection as well as a CLOB element to capture additional information.

Process Flow's Business Object Controls Everything

A process flow's business object controls its contents, lifecycle and various other business rules:

- Its schema defines where each piece of information resides on the physical Process Flow maintenance object.
- It may define a lifecycle for all process flow instances of this type to follow. Each process flow must exist in a valid state as per its business object's lifecycle definition.

- It may define validation and other business rules to control the behavior of process flows of this type.

Using Process Flow Type To Control Process Flows

Each process flow must reference a process flow type. The process flow type will reference business objects that define:

- The structure and processing options of the process flow type itself.
- The structure, lifecycle, processing rules and processing options of the actual process flow.

The process flow type may be used to define parameters for the associated process. Examples of such parameters are:

- Account Types, Obligation Types or Tax Types applicable to the process.
- Adjustment Types for any financial transactions that are created as part of the process.
- Customer Contact Classes / Types for any correspondence triggered from the process flow.

Person / Account / Tax Role / Obligation References

Some types of processes may be person-oriented, others may be obligation-oriented or tax role-oriented, and still others may be account-oriented. Any combination of person, account, tax role and obligation is permitted on a process flow. The associated business object may be designed to include only those objects that are relevant to the particular type of process being modeled.

Process Flow Supports A Log

The Process Flow maintenance object supports a log. Any significant event related to a Process Flow may be recorded on its log. The system automatically records a log record when the process flow is created and when it transitions into a new state. In addition, any custom process or manual user activity can add log entries.

Access Rights

You can take advantage of the system's business object security functionality to restrict processes of a given type and states within the process lifecycle to certain user groups.

Additional Documentation

For more information about the various configuration tools available to customize the process flow maintenance object, refer to the Configuration Tools documentation.

Refer to *The Big Picture of Business Objects* for details of how to configure content, lifecycle and business rules. Refer to *State Transitions Are Audited* for more information on business object logs. Refer to *Granting Access To Business Objects* for more information on security options.

For more information about user interfaces, refer to *the Configurable User Interface* documentation.

Setting Up Process Flows

The following sections describe how to prepare process flow types.

Setting Up Process Flow Types

Each process flow requires a process flow type. To set up a process flow type, select **Admin Menu** > **Process Flow Type**.

The topics in this section describe the base-package zones that appear on the Process Flow Type portal.

Process Flow Type List

The Process Flow Type List zone lists every process flow type. The following functions are available.

- Click a broadcast button to open other zones that contain more information about the adjacent process flow type.
- Click the **Edit** button to maintain the process flow type.
- Click the **Duplicate** button to copy the process flow type.
- Click the **Delete** button to delete the process flow type.
- Click the **Add** link in the zone's title bar to add a new process flow type.

Actions

This is a standard actions zone. The **Edit, Delete** and **Duplicate** actions are available.

Process Flow Type

The Process Flow Type zone contains display-only information about a process flow type. This zone appears when a process flow type has been broadcast from the Process Flow Type or if this portal is opened via a drill down from another page.

The UI Map used for the display is determined by the Display UI Map option on the associated business object.

The base package does not provide a business object for Process Flow Type. Each implementation must define its own business objects and user interfaces.

Defining Case Options

The Case functionality was originally provided as a highly configurable tool for managing business processes in situations where the base product didn't include a dedicated maintenance object. This functionality has now been superseded by either specific functionality (e.g. Appeals, Bankruptcy), or by the use of *Process Flow*.

The topics in this section describe how to configure the system to manage cases for those implementations that are continuing to use the legacy case functionality.

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 cases.

Case Type Controls Everything

Whenever a user creates a case, they must specify the type of case. The case type controls how the case is handled.

Case types hold the business rules that control cases. Since these business rules can sometimes be quite complicated, setting up case types requires planning and foresight. The topics in this section describe the type of business rules that can be configured on your case types.

Person / Account / Location Applicability

Some types of cases may be person-oriented, others may be location-oriented, and still others may be account-oriented. When you set up a case type, you define if its cases must reference a person, account, and/or location. Note, any combination of these objects is permitted on a case.

Contact Information Applicability

When a case is created as a result of contact from an outside party, 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*). 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. 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.

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 lifecycle whose rules are governed by the base-package and cannot be changed. For example, an obligation starts out in the **Pending Start** state and eventually becomes **Closed** when it's been final billed and/or completely paid. You can't change the system to allow an obligation to starts its life in the **Closed** state.

The lifecycle of cases is not governed by the base product. Rather, you define the lifecycle of your cases when you set up their case types.

The topics that follow describe important concepts:

Valid States versus State Transition Rules

A given case can have one or more potential valid states. State transition rules govern the states a case can move to while it's in a given 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. This means that a user will never see the case in this state. If the other states were marked as **non-transitory**, and an error were to occur during the transition from a transitory state, the case would roll back any changes to data made in the entered state (Enter Processing) along with the changes made in the transitory state, and would end up in the last non-transitory state prior to the transitory 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.

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.

The "final" states are used by the system to differentiate between open and closed cases. For example, an alert highlights when the person / account / location 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).

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 Allow A User 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. The case will transition to the appropriate state depending the button the user presses.

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. 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. 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.
- Rather than make a user launch a script by pressing a hyperlink on the *case page*, you can have the system automatically launch the script while the case is in a given state. Refer to *Script Launching Option* for more information.
- You can also have the system automatically launch a script when a user selects a To Do entry. Refer to *Launching Scripts When To Do Entries Are Selected* for more information.

Fastpath:

Refer to *Scripts and Cases* for more information about how to streamline your case processing with scripts.

Required Fields Before A Case Enters A State

You can define additional fields (i.e., characteristics) that are required before a case can enter a given state. For example, You can indicate a case must reference a cancel reason before it enters a **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 **Assigned** 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 *To Do entry* created when a case enters a given state. This additional processing is held in algorithms that are plugged in on the case type.

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. Note that your Exit Validation and Exit Processing logic, if configured for the case state, will still be executed as part of the state transition. Auto-Transition logic for this state will be ignored during this transition.

Validation Before A Case Exits A State

You can define validation that executes before a case can exit a given state. For example, you might want to check the account's balance is less than a given value before a collection 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 case leaves a certain 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 using auto transition rules. For example, if you have a case that sends a letter to a taxpayer, it can be configured to transition to the **Follow Up** state 1 week after the letter 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.

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** 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. 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 formal appeal.

Using a script to create a case can save a user a lot of time (and training efforts). This is because the script can automatically populate many fields on the case based on answers to questions.

Refer to *Initiating Scripts* for a description of how end-users initiate scripts.

- You can create a script to help users work on a case when it's in a given state. Refer to *A Script That Helps A User Work A Case* for more information.
- You can *set up your case types to create To Do entries* to notify users when cases exist that require their attention. Users can complete many of these To Do entries without assistance. However, you can set up the system to

automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a formal appeal that requires investigation. You can set up the system to execute a specific script when a user selects this To Do 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 functionality 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. To implement this, you can set up the case type as follows:

- Plug-in an *entry processing* algorithm on the status where you wish a To Do entry to be created or completed.
- Plug-in an *exit processing* algorithm on the state where you wish the To Do entry to be created or completed.

Launching Scripts When To Do Entries Are Selected

You can set up your case types to create To Do entries to notify users when cases exist that require their attention. Users can complete many of these To Do entries without assistance. However, you can set up the system to automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a formal appeal that requires investigation. You can set up the system to execute a specific script when a user selects this type of To Do entry. This script might guide the user through the investigation process. Refer to *Executing A Script When A To Do Entry Is Selected* for more information.

All To Do Entries Are Visible

When a case is displayed on *Case Maintenance*, the system summarizes the number of To Do entries associated with the case (if you've *set up your To Do types* appropriately).

Setting Up Case Options

The topics in this section describe how to set up the system to enable case functionality.

⚠ **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

The case information displayed throughout the system is controlled by a plug-in.

The system first looks to see if the case type references a case business object and if the business object defines an information plug-in. If a BO is not provided or if that BO does not define an information algorithm, the system looks for an information algorithm plugged into the case maintenance object.

If no plug-ins are found on the BO or the MO, the system looks for a plug-in algorithm on the *Case Type*.

If such an algorithm is not plugged-in on the Case Type, the system looks for a corresponding algorithm on the *installation record*.

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 / location in context has open cases. Refer to *CCAL-CASE* for an example of this algorithm.

Setting Up Application Services

As described under *Access Rights*, you can prevent unauthorized users from accessing cases. The following points describe how to implement this type of security:

- Create an *application service* for each case type that needs to be secured
- Create an action on the application service for each status you need to secure
- Link the valid *user groups* to the application service and define which actions they can perform
- Define the application service on the *case type*
- Define the related action for each status on the *case type / status*

Setting Up Scripts

As described under *Scripts and Cases*, BPA scripts can facilitate the creation and working of cases. Refer to the *Defining Script Options* for instructions describing how to set up scripts.

Setting Up To Do Types

As described under *To Do's and Cases*, To Do entries can be used to highlight cases that require user attention.

The following points provide a high-level description of how to create (and complete) To Do entries for a case type:

- Create a To Do type for each different type of To Do entry used during a case's lifecycle
- On the To Do type, think carefully about the roles whose users can work on the entries
- Also consider if you would like a BPA script launched when a user selects the entry
- Specify the To Do type on the appropriate *entry processing* or *exit processing* algorithm
- If you want the system to automatically complete To Do entries, specify the To Do type on the appropriate *entry processing* or *exit processing* algorithm

Please be aware that the case maintenance transaction highlights the number of open and being worked To Do entries linked to the case being displayed on the page. However, the system can only do this if the To Do entries reference a *foreign-key characteristic* whose foreign key references the case table. If you use the *CSEN-TD* algorithm to create To Do entries when a case enters a given state, this algorithm will do this for you if:

- You have set up a *foreign-key characteristic type* whose *foreign key* references the case table
- In addition, the characteristic type must reference a characteristic entity of **To Do Entry**

Setting Up Characteristic Types

As described under *Additional Information*, some of your cases may require additional information (in the form of *characteristics*). If this is true, you must set up the characteristic types before setting up the case types.

Refer to *Setting Up To Do Types* for instructions regarding a characteristic type that must be set up in order for the system to know the To Do entries that are associated with a case.

If you use the *CSEN-CC* algorithm to create customer contacts when a case enters a given state, you should set up a *foreign-key characteristic type* as follows:

- Its *foreign key* must reference the case table
- In addition, the characteristic type must reference a characteristic entity of **Customer Contact**

Setting Up Case Types

The case type maintenance transaction is used to maintain your case types. The topics in this section describe how to use this transaction.

Fastpath:

Refer to *The Big Picture Of Cases* for more information about how a case type encapsulates the business rules that govern a case.

Case Type - Main

Use this page to define basic information about a case type.

Open the case type page by selecting **Admin Menu** > **Case Type**.

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.

Location Usage controls the applicability of a location on cases of this type. Select **Required** if a location must be defined on this type of case. Select **Optional** if a location can optionally be defined on this type of case. Select **Not Allowed** if a location 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" may be constructed using an algorithm plugged in here. Refer to *Case Info May Be Formatted By An Algorithm* for more information on when this algorithm is used. 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 lifecycle. Before you can design a case type's lifecycle, it's important that you thoroughly understand the concepts described under *Lifecycle* and *Status-Specific Business Rules*. After thoroughly understanding these concepts, we recommend you perform the following design steps:

- Draw a "state transition diagram" as illustrated above under *Lifecycle*. Keep in mind that if your state transition diagram is complex, your cases will be complex. While some cases warrant complexity, you should always ask yourself if there aren't better ways to achieve the desired results if your first effort results in complexity.
- Determine which characteristics (if any) are required during each stage of a case's lifecycle
- Determine when To Do entries (if any) should be created (and completed) during a case's lifecycle
- Determine additional validation (if any) that should be executed before a case enters and exits each state

- Determine additional processing (if any) that should transpire when a case enters or exits each state
- Determine if scripts are warranted to help users work the cases and, if so, design the scripts for each applicable state

When the above tasks are complete, you will be ready to set up a case type's lifecycle.

Open the Lifecycle page by selecting **Admin Menu** > **Case Type** and navigate to 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 taxpayers with cases in the state. (The alert is shown via the base package Installation - Alert algorithm.)

Algorithms

The **Algorithms** grid contains algorithms that control important functions for cases of this type. You must define the following for each algorithm:

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

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	***Description***
Auto Transition	This algorithm is executed to determine if a case that's in this state should be transitioned into another state. Refer to *Automatic Transition Rules* for the details. Click *here* to see the algorithm types available for this system event.
Enter Processing	This algorithm holds additional processing that is executed when a case is transitioned into this state. You can also specify state transition logic within Enter Processing routines. Refer to *Additional Processing When Entering A State* for the details. Click *here* to see the algorithm types available for this system event.
Enter Validation	This algorithm holds validation logic that executes before a case can enter a given state. Refer to *Validation Before A Case Enters A State* for the details. Click *here* to see the algorithm types available for this system event.
Exit Processing	This algorithm holds additional processing that is executed when a case is transitioned out of this state. Refer to *Additional Processing When Exiting A State* for the details. Click *here* to see the algorithm types available for this system event.
Exit Validation	This algorithm holds validation logic that executes before a case can 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.

Reports Addendum

This chapter is an addendum to the general *Defining and Designing Reports* chapter. This addendum describes the sample reports that are provided with Oracle Enterprise Taxation Management.

Description of Sample Reports

This section provides an overview of each sample report supplied with Oracle Enterprise Taxation Management that may be found in the demonstration database. They may be used by your organization as they are or as a starting point for creating a *new report*.

Note:

Account Security. Please note that the sample reports provided with the product do NOT incorporate account security. If a user has been given security to view the report, then all the data in the report is available for viewing.

Billed Revenues by Rate - CI_BILREV

Parameters

Parameter	*Parameter Code*	*Description*
Accounting Period	P_ACCT_PERIOD	Defines the accounting period used for the report. A valid fiscal year and accounting period for a valid accounting calendar must be provided.

Account type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an account type.
Account type	P_REV_ACCTY_CHAR	Account type Char Value for Revenue related GL Accounts. The char type defined for this parameter should match the Char Type code defined as parameter #2. The parameter value indicated for this parameter should be one that represents revenue accounts.

Report Description

This is an analysis report for the billed revenues for an accounting period according to the various rates, which were in effect in the system. The information in this report helps to adjust rates in order to achieve better financial results and comply with regulations and market trends.

This report selects all records in the financial transaction GL collection that satisfy the following criteria:

- The financial transaction is frozen.
- The Accounting Date on the financial transaction within input accounting period (parameter 1)
- The distribution code associated with the GL entry has a characteristic type and value that matches the input account type Characteristic and account type (parameters 2 & 3)

Note:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. Note that many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Case Statistics By Case Type - CI_CSESTS

Parameters

Parameter	***Parameter Code***	***Description***
Start Date	P_FROM_DT	See the report's description for how this field is used. If start date is not specified, it is defaulted to 7 days prior to the end date.
End Date	P_TO_DT	See the report's description for how this field is used. If end date is not specified, it is defaulted to the current processing date.
Case Condition (Open, Closed)	P_COND_FLG	If specified, only cases in this condition are included in the report. If left blank, the reports produces statistics for both open and closed cases.

Report Description

This report provides two types of statistics:

1. Open cases whose creation date falls between the input Start Date and End Date (inclusive)

2. Closed cases whose closing date falls between the input Start Date and End Date (inclusive)

The third parameter is only used if you want to restrict the statistics to only open or closed cases. If you leave this parameter blank, both open and closed statistics will be produced.

The following information is provided in graphical format:

- Number of cases by case type
- Percentage of cases by case type

Case Statistics for a Given Status - CI_CSESGS

Parameters

Parameter	***Parameter Code***	***Description***
Start Date	P_FROM_DT	See the report's description for how this field is used. If start date is not specified, it is defaulted to 7 days prior to the end date.
End Date	P_TO_DT	See the report's description for how this field is used. If end date is not specified, it is defaulted to the current processing date.
Case Type/Status	P_CASE_STATUS_CD	This is the desired Case Type and Status that will be reported on.
Responsible User	P_CASE_OWNER	If specified, only cases with this responsible user are included in the report.
First Bucket High Limit	P_B1_LIMIT	Cases that took <= this number of days to reach the given status will be grouped together for statistical reporting.
Second Bucket High Limit	P_B2_LIMIT	Cases that took <= this number of days but more than the first bucket high limit to reach the given status will be grouped together for statistics reporting.
Third Bucket High Limit	P_B3_LIMIT	Cases that took <= this number of days but more than the second bucket high limit to reach the given status will be grouped together for statistics reporting. Cases that took more than this number of days are included in another group.

Report Description

This report shows cases of a given case type that transitioned to a given status during a given date range.

Graphs are printed to show the number and percentage of cases grouped by the time it took to reach the status. These statistics are grouped into age buckets whose size is controlled by the last 3 parameters.

Summary statistics are also printed showing the minimum, maximum, average and median times for these cases.

Customer Contacts by Type - CI_CUSTCN

Parameters

Parameter	*Parameter Code*	*Description*
Start Date	P_FROM_DT	Start date to use for reporting customer contacts. If not defined, the start date is set to the current date minus 7 days.
End Date	P_TO_DT	End date to use for reporting customer contacts. If not defined, End Date is set to the current date.
Customer Contact Class / Type	P_CC_TYPE_CD	Specify a Customer Contact Class / Type to restrict the report output to a specific class / type.

Report Description

This report lists all customer contacts in the system created within the input date range. You may optionally restrict the report to customer contacts for a given Customer Contact Class / Type (parameter 3).

Note:

Graphs. The information on this report is shown in both textual and graphical formats.

Note:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_CC table on CC_DTTM to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. Note that many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

GL Accounting Summary - CI_GLACSM

Parameters

Parameter	*Parameter Code*	*Description*
Accounting Period	P_ACCT_PERIOD	Defines the accounting period used for the report. A valid fiscal year and accounting period for a valid accounting calendar must be provided.
Account Type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an account type. The account types for the GL accounts are used for grouping the output to the report.

Report Description

This is a financial audit report used to check the financial details in Oracle Enterprise Taxation Management for an accounting period against the GL system. The report summarizes all financial transaction (FT) information for a given

accounting period according to the different operating and GL divisions and according to various levels of the account GL information.

Note:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. Note that many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Open Cases By Type - CI_CSEOPN

Parameters

Parameter	*Parameter Code*	*Description*
Start Date	P_FROM_DT	See the report's description for how this field is used. If start date is not specified, it is defaulted to 7 days prior to the end date.
End Date	P_TO_DT	See the report's description for how this field is used. If end date is not specified, it is defaulted to the current processing date.
Case Type	P_CASE_TYPE_CD	If specified, only cases of this type are included in the report.
Responsible User	P_CASE_OWNER	If specified, only cases with this responsible user are included in the report.
First Bucket High Limit	P_B1_LIMIT	Cases that are open for less than or equal to this number of days will be grouped together for statistical reporting.
Second Bucket High Limit	P_B2_LIMIT	Cases that are open less than or equal to this number of days but more than the first bucket high limit will be grouped together for statistics reporting.
Third Bucket High Limit	P_B3_LIMIT	Cases that are open less than or equal to this number of days but more than the second bucket high limit will be grouped together for statistics reporting. Cases that took more than this number of days are included in another group.

Report Description

This is a report on open cases that were created between a given date range.

The report can be limited to a specific type and/or responsible user.

For each case type, the report shows the following:

- Number of open cases by age bucket (the last 3 parameters control the size (in days) of each bucket)

- Percentage of open cases by age bucket
- Details of the open cases

Payments Balance - CI_PMTBAL

Parameters

Parameter	*Parameter Code*	*Description*
Start Date	P_FROM_DT	Report gets all the payments that have been received during a given date range (from and to date parameters). If start date is not defined by the user, it is set to 7 days prior to the current date.
End Date	P_TO_DT	End date of the date range. If not defined by user it is set to the current date

Report Description

This report provides an overall view of all payments created within the input date range. It is typically used for financial control and audit purposes. The report provides summary information about valid payments received and about canceled payment. Data is summarized by the tender source and the type of payment.

Receivables Aging - CI_RCVAGA

Parameters

Parameter	*Parameter Code*	*Description*
Cutoff Date	P_CUTOFF_DATE	The date from which the arrears buckets are calculated. If no value is entered, the default is the current date minus 7 days.
1 st Bucket High Limit	P_B1_LIMIT	High limit of 1st bucket.
2 nd Bucket High Limit	P_B2_LIMIT	High limit of 2 nd bucket.
3 rd Bucket High Limit	P_B3_LIMIT	High limit of 3 rd bucket.

Report Description

This report lists all accounts and their arrears information as of the input cutoff date using a balance forward accounting method.

Outstanding debt is placed into the buckets provided as input using the age of the debt as of the cutoff date. Credits are applied to the oldest debt first. For each account a separate bucket is used to display new charges. In addition, the total accounts receivable balance is displayed for each account.

Note:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ARS_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. Note that many companies opt to create a reporting database that

is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

Tax Payables Analysis - CI_TXPYBL

Parameters

Parameter	*Parameter Code*	*Description*
Start Date	P_FROM_DT	Show summary of the tax amounts starting from this date. If not specified, the system will default this value to the current date minus 7 days.
End Date	P_TO_DT	Show summary of the tax amounts up to this date. If not specified, the system will default this value to the current date.
Account type Characteristic	P_CHAR_TYPE	Defines a Characteristic Type for a characteristic linked to the Distribution Code to define an account type.
Account type	P_TAX_ACCTY_CHAR	Account type Char Value for tax related GL Accounts. The char type defined for this parameter should match the Char Type code defined as parameter #3. The parameter value indicated for this parameter should be one that represents tax liability accounts.

Report Description

This report displays a summary of the tax amounts that were levied by the company to taxpayers within the input date range. It also includes the tax exemption information for that period.

This report select all records in the financial transaction GL collection that satisfy the following criteria:

- The financial transaction is frozen.
- The Accounting Date on the financial transaction within the input date range
- The distribution code associated with the GL entry has a characteristic type and value that matches the input account type Characteristic and account type (parameters 3 & 4)

The report also provides tax exemption information for bill segments whose financial transactions satisfy the above criteria. The tax exemption information is retrieved by looking at the bill calculation lines associated with the FT's bill segment.

Note:

Performance Consideration. If your implementation chooses to use this report, you may consider adding an index to the CI_FT table on ACCOUNTING_DT to aid in performance. When making this decision, carefully weigh the benefit of improving report performance against the possible degradation to the performance of day-to-day processing as a result of defining a new index. Note that many companies opt to create a reporting database that is a shadow of production to ensure that indexes defined to benefit reports may be created without any affect on the production environment.

To Do Entries - CI_TDENTR

Parameters

Parameter	Parameter Code	Description
ToDo Entry Status	P_ENTRY_STATUS_FLG	Defines if the ToDo entries on the report should be limited to those with a given status value. If this parameter is left blank, the report will show all **open** and **being worked** ToDo entries.
ToDo Type	P_TD_TYPE_CD	Defines if the ToDo entries on the report should be limited to those of a given ToDo type. If this parameter is left blank, the report will show all ToDo type that have at least one **open** or **being worked** entry.

Report Description

The report shows open and being worked To Do entries.

You can limit the report to entries in a given status by specifying the desired **To Do Status** (open or being worked). If you don't specify a status, all **open** and **being worked** To Do entries will appear on this report.

You can also limit the report to entries of a given To Do Type by specifying the desired **To Do Type**. If you don't specify a To Do Type, all To Do Types with at least one entry in the **open** / **being worked** state will appear on this report.

Graphs. The information on this report is shown in both textual and graphical formats.

Security Addendum

This chapter is an addendum to the general *Defining Security and User Options* chapter. This addendum describes security functionality that is specific to Oracle Enterprise Taxation Management.

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 taxpayers in two geographic territories, you may need to restrict access to accounts based on the office that manages each territory. For example, only users in the northern office may manage accounts in the northern territory.
- If you have businesses and individual taxpayers, you may need to restrict access to these different taxpayer segments based on the skill set of the users. For example, some users are skilled in dealing with business taxpayers, while others are skilled in dealing with individual taxpayers.

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

Fastpath:

Refer to *If You Do Not Practice Account Security* for setup instructions if your organization doesn't practice account security.

Note:

Account security may also affect persons and locations. Refer to *Persons Can Also Be Secured* for how access to person information is also restricted by account security. Refer to *Locations Can Also Be Secured* for how access to location 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 method described under *How To Add A New Taxpayer From Control Central*. In addition, you may manually link and unlink persons from account using the *Account - Person* page.

Locations Can Also Be Secured

It's important to be aware that locations can also be secured as a result of "account security". It works like this:

- If a location is linked to at least one account, users will not be allowed access to the location (or the location's related information) unless they have access to at least one of the location's accounts.
- If a location is not linked to an account (a rare situation), then all users may access the location.

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 *Control Central*:

- Searches are affected as follows:
- An account will only be visible if a user has access to the account's access group.
- Persons that are not linked to accounts will be visible to all users.
- If a person is linked to an account, the person will only be visible if the user has access to at least one of the person's accounts.
- Locations that are not linked to accounts will be visible to all users.
- If a location is linked to an account, the location will only be visible if the user has access to at least one of the location's accounts.
- The alerts that highlight the existence of "multiple relationships" are not impacted by account security. Specifically:
- The alert **Person has multiple accounts** will appear if the selected person is linked to multiple accounts, even if the user doesn't have access to every account. Note well, the person couldn't have been selected if the user didn't have access rights to at least one account.
- The alert **Location has multiple accounts** will appear if the selected location is linked to multiple account, even if the user doesn't have access to every account. Note well, the location couldn't have been selected if the user didn't have access rights to at least one account.
- Only accounts to which the user has access will be displayed in the person tree.
- Only accounts to which the user has access will be displayed in the account tree.

- All other pages contain information related to Control Central's current account context. The current account context can never reference an inaccessible account and therefore these pages are not impacted by account security.

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 obligations that are, in turn, associated with accounts to which they have access rights.

Note:

Person and location searches are also impacted. Keep in mind that information will be suppress from both person and location-oriented searches if the person / location is related to accounts. Refer to *Persons Can Also Be Secured* for how access to person information is also restricted by account security. Refer to *Locations Can Also Be Secured* for how access to location information is also restricted by account security.

Account Security and To Do Lists

Account security does NOT impact the information that appears in a user's To Do list. Rather, we have assumed that your To Do roles (and the users assigned to these roles) are consistent with your account security requirements. This can result in anomalies. For example, it's possible for a supervisor to assign a bill segment error to a user who doesn't have access to the bill segment's account. This user will then see the related To Do entry in their Bill Segments In Error To Do list. However, when they drill down on the entry, account security will manifest itself (i.e., the user won't be able to display the bill segment that's in error). This happens because the drill down causes the bill segment search logic to execute. This logic inhibits the selection of bill segments if the user can't access the related account.

To minimize these anomalies, we recommend the following:

- Setup *To Do Roles* consistent with your Data Access Roles.
- Setup *Account Management Groups* that are consistent with your Access Groups.
- Setup default To Do Roles on your Account Management Groups for each *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.
- **Location-oriented.** This notation is used if the respective transaction uses location-oriented account security. Refer to *Locations 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

Transaction	*Type of Account Security*
Account Financial History	Account-oriented
Account Payment History	Account-oriented
Account Person Replicator	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
Case	Account-oriented, Person-oriented and Location-oriented
Collection Agency Referral	Account-oriented
Control Central	Account-oriented, Person-oriented and Location-oriented
Customer Contact	Person-oriented
Financial Transaction	Account-oriented
Financial Transaction on a Bill	Account-oriented
Financial Transaction on a Payment	Account-oriented
Match Event	Account-oriented
Multi-Cancel/Rebill	Account-oriented
Overdue Process	Account-oriented
Pay Plan	Account-oriented
Payment	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

Transaction	*Type of Account Security*
Location	Location-oriented
Obligation Billing History	Account-oriented
Obligation Cash Accounting Balance	Account-oriented
Obligation Financial History	Account-oriented
Obligation	Account-oriented

Account Security Case Study

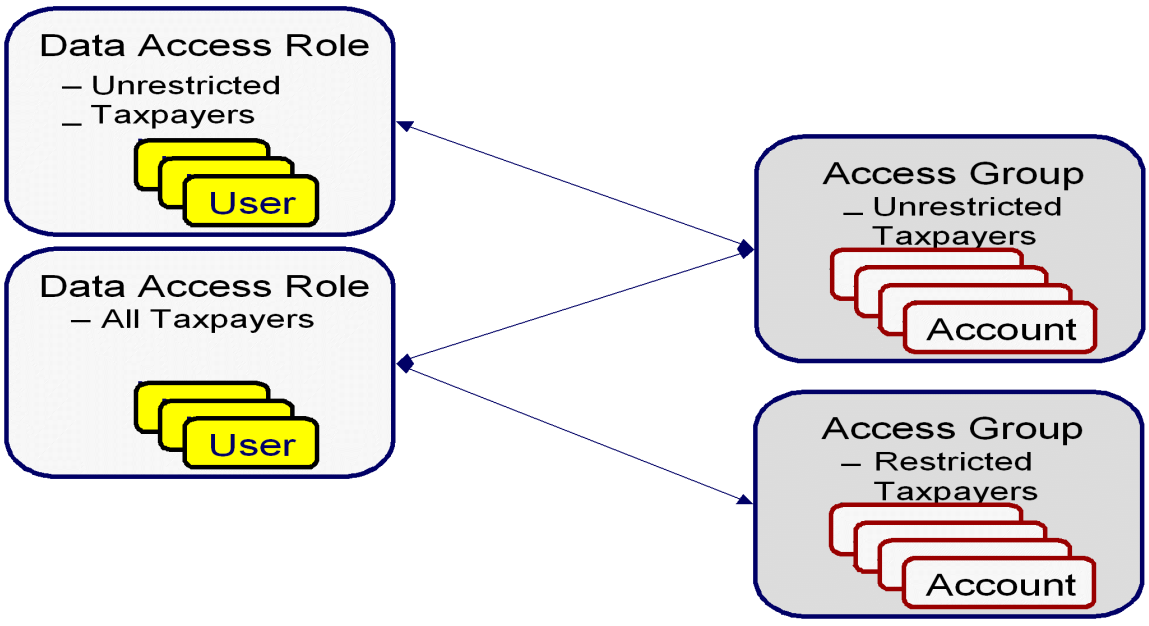
This section contains an example of how to implement account security. Use this example to form an intuitive understanding of these objects. Once this intuition is obtained, you'll be ready to design the account security objects for your own company.

Securing Accounts Based On Account Type

Assume the following security requirement exists:

- You have two broad groups of accounts:
- Unrestricted Taxpayer accounts for the general public.
- Restricted Taxpayer accounts for individuals whose tax information is highly sensitive (politicians, celebrities, employees of the tax authority, etc.).
- Users can be classified as have one of the following access rights:
- May access all accounts.
- May only access the Unrestricted Taxpayer 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 two access groups because access to accounts is based on whether the taxpayer's account is unrestricted or restricted.

- The Unrestricted Taxpayers data access role is only linked to the Unrestricted Taxpayers access group.
- The All Taxpayers data access role is linked to both the Unrestricted Taxpayers and Restricted Taxpayers access groups. Users with this role can therefore access all accounts.

The Default Access Group

The base package defaults an account's access group based on the user who adds the account. It uses the user's *default access group* to do this.

Caution:

Please be aware that the user who adds an account must have access to this access group.

Note:

Subsequent changes to an account's access group. A user may change an account's access group to any access group to which they have access.

If You Do Not Practice Account Security

If you do not restrict access to accounts (i.e., all users can access all accounts), you must set up one access group and one data access role and then indicate all users are part of this role. You should also define the access group as the default access group on all of your users (so that new accounts are all labeled with this access group).

Masking Sensitive Data

Refer to *Masking Data* for instructions describing how to configure the system to mask sensitive data like a taxpayer's social security number or bank account number. If your implementation intends to mask any of the information that appears in the *Taxpayer Information Zone*, please navigate to this zone's documentation for special instructions.

Inquiry Audit

There are times when an organization needs to capture audit information when a user views certain information. Examples of these scenarios include:

- A user is under investigation or disciplinary action, and supervisors need to find out what information the user has been looking at.
- Although VIP accounts may be protected from users by account security, supervisors and fraud investigators may still want to know who has been trying to look for forms or other records with a given criteria.
- Sometimes information about a taxpayer is leaked to the newspaper. For example "Town Councilman Smith didn't pay his taxes on time for the last n years". Supervisors want to know who has recently viewed Councilman Smith's tax returns.

The audit information captured includes:

- What - the specific record that was viewed.
- Who - the user viewing the record.
- Where the user performed the inquiry - which portal, zone or page was used.
- How - which search criteria the user entered for a specific query.

For more on the information provided for the audited records, see *Inquiry Audit Query*.

As part of implementing inquiry audit, you will need to decide which portals, zones and pages require inquiry audit. The following section describes how to configure inquiry audit based on how the page or portal is implemented.

Setting Up Inquiry Audit

The topics in this section describe how to set up inquiry audit for the various types of queries.

Setting Up Audit for Portals and Zones

The zone parameter **Audit Service Script** controls if information about the record being inquired upon will be captured for audit. In order to turn on inquiry audit for a zone, you will need to update the **Override Parameter Value** with the following:

- The name of the service script responsible for capturing the audit data and storing it on the Inquiry Audit table.
- The base includes the service script **C1-ZoneAudit**
- Additional scripts can be developed if your implementation requires something more specific such as add audit records only if VIP records are viewed or where certain criteria is used.
- The mnemonic **ss=** is used to define the name of the service script. For example ss='C1-ZoneAudit'.
- Define the input elements to the service script by using the mnemonic**input=**. These can include the user id, zone, portal, user filter values, hidden filter values, any literal value, any portal or global context field.

See the zone's help on more information.

Setting Up Audit for a Search Page

A **user exit** is needed to turn on inquiry audit for an old style search page or control central. You will need to do the following:

- Determine the service name of the page.
- Create a search page extension to capture the audit information. . The following extension codes are provided as samples:
- **ext_ multiSearchNameData.jsp** - Control Central Search by Name
- **ext_ multiSearchAddressData.jsp** - Control Central Search by Address
- **ext_ accountSearchData.jsp** - Account Search
- The search pages have grid elements that display the result rows. To capture the search criteria, the service script **C1-PageAudit** should be called upon load of the grid element, to add the inquiry audit record.
- The business object to use when calling the service script is **C1-PageAuditInquiry**

Setting Up Audit for a Maintenance Page

A **user exit** is needed to turn on inquiry audit for an old style maintenance page. You will need to do the following:

- Determine the service name of the page.
- Create a page maintenance page extension to capture the audit information. . The following extension code is provided as a sample:
- **CMAccountMaintenanceExtension.java** - Account Page Maintenance
- The page maintenance extension method **afterRead** should be overridden to call the service script **C1-PageAudit** to add the inquiry audit record.
- The business object to use when calling the service script is **C1-PageAuditInquiry**

Inquiry Audit Restrictions

Certain portals are designed to include a zone with a list of records and buttons that allow quick editing of a record. This technique invokes a BPA script that displays the data in a UI map, rather than broadcasting a value from a query or info zone. This bypasses the ability to capture the audit for this record. There is currently no ability for an implementation to include a user exit in a BPA script to allow storage of an audit record.

The above applies to any administrative portal that uses the "all in one" portal design that includes a list zone showing all existing records. The list zone typically includes edit, duplicate and delete buttons.

Note For customized portals that call a customized BPA script, an implementation has the ability to include a step to store an Inquiry Audit record.

Inquiry Audit Query

Use the *query portal* to search for the inquiry audit records.

The inquiry audit search allows you to search for inquiry records using various criteria such as:

- Date that the inquiry occurred.
- Portal and/or Zone - you can search on inquiries made in a specific portal or a specific zone.
- Service name (used for page inquiries)
- User or User Group - you can search on inquiries made by a specific user or see inquiries made by a group of users.
- The object inquired on, e.g. Account.
- Specific field inquired on, e.g. Name
- Specific field value used in the inquiry. For example, you can look at inquiries made for a specific name, a certain amount, a specific address.
- A combination of the above criteria.

Once a inquiry audit record is selected from the list, the information displayed depends on the type of inquiry that was performed and may differ based on the business object used for capture. Refer to the zone's help text for more information.

Defining Overdue Processing and Collection Options

The system periodically monitors how much your taxpayers owe to ensure they haven't violated your collection rules. When a violation is detected, the system initiates the appropriate activities (e.g., letters, collection agency referrals, and eventually write off). The topics in this section describe how to configure the system to manage your overdue processing and collection requirements.

Note:

Collecting on unpaid debt. 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.

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 taxpayer, the age of debt, the type of tax, etc.), your setup process will be more challenging.

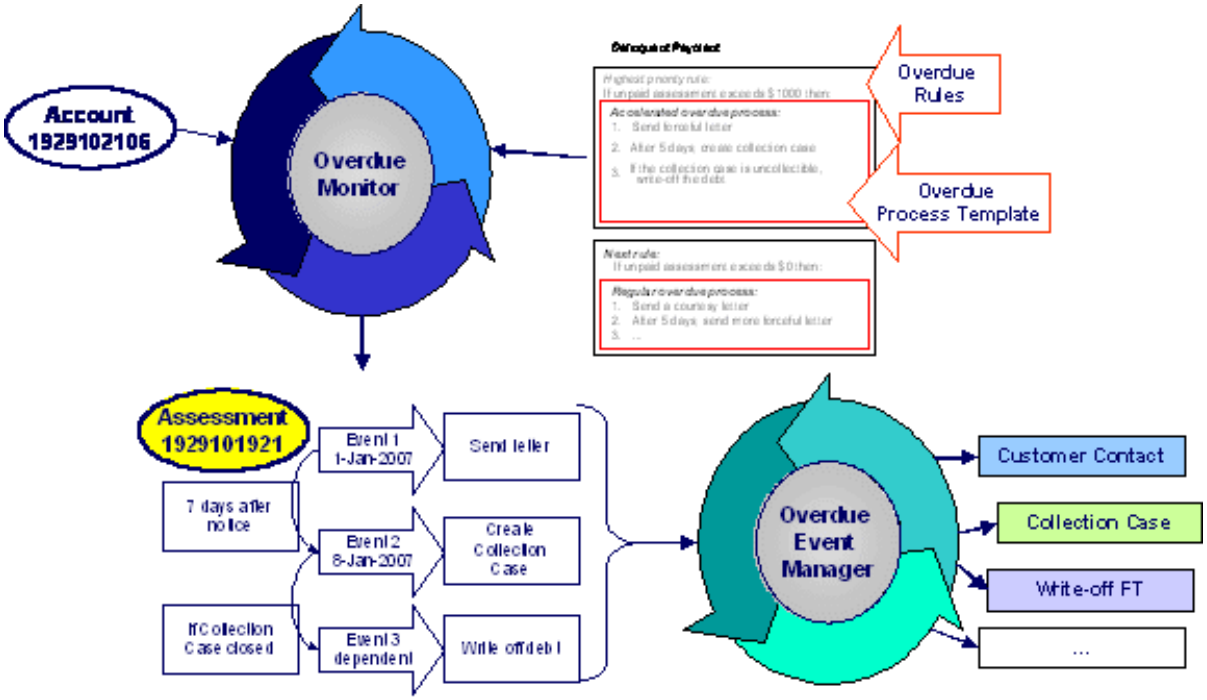
Note: The terms "customer" and "taxpayer" may be used interchangeably in this chapter.

Case Study - Collecting On Overdue Debt

The following topics introduce a case study that describes how overdue processing can be used to collect on overdue debt. 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 Debt

The following diagram illustrates the objects and processes involved with collecting overdue debt.



There are many important concepts illustrated above:

The Overdue Monitor checks if your accounts have debt that violates 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. Collection classes are monitored by the Overdue Monitor. This chapter describes the Overdue Monitor.
Overdue rules define when and how unpaid debt is 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. When you set up a monitoring algorithm, you define the type of overdue process that should be created when overdue debt is detected. You do this by defining the appropriate "overdue process template".
An overdue process template defines how to handle overdue debt	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 taxpayer 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 product.
Multiple objects can be associated with a single process	The above diagram shows a single assessment linked to an overdue process. It should be noted that an overdue process is capable of referencing multiple assessments (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 obligations under a single overdue process. The type of object managed by an overdue process is defined on its *overdue process template*.
If a taxpayer pays the debt, the overdue process is cancelled	If the overdue debt 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 state (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.

Recommendation. We recommend that you familiarize yourself with the concepts described in the *case studies* before reading this section.

Contents

Different Overdue Rules For Different Taxpayers

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 may have different collection rules for different jurisdictions (i.e., divisions).
- You may have different collection rules for different taxpayer segments. You differentiate your taxpayers in respect of the overdue via the **collection class on the taxpayers' accounts**. An account's initial collection class is defaulted from its account type. You may override an account's collection class at will.
- You may have different criteria for every currency in which you work because the monitoring process always compares a taxpayer's debt against some value and this value must be denominated in the taxpayer's currency. A taxpayer's currency is defined using a **currency code on the account**.

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*

Note:

Additional rules. Your implementation can have an **Overdue Monitor Rule**that caters for credit balances on obligations. For an example of an algorithm that creates an overpayment process for an obligation in credit, please refer to the base algorithm type *C1-CC-INITOP*.

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 *Account Type - Controls* associated with the account's account type and division (in the field Min Compliance Review Freq (Days)).

In addition, the Overdue Monitor examines an account's debt 'on demand' by looking at a special table called Compliance Review Schedule, which contains the account ID and a review date. Processes can insert a record into this table if something happens to the account. Some examples of base processes that insert a row in this table include:
- Bill completion. A row is inserted for the account for the bill due date plus the Compliance Review Grace Days specified on *account type control*.
- FT freeze. A row is inserted for the account for the current date if the obligation's balance is considered overdue or for the payment due date plus the Compliance Review Grace Days specified on account type if the obligation's balance is not yet due.
- If a payment is canceled with a cancellation reason that indicates non-sufficient funds.
- If a *match event* is added, changed or deleted.

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 Compliance Review Schedule table (CI_ADM_RVW_SCH) when the event occurs populating the account ID and an appropriate review date. Note that the Compliance Review Grace Days on the account type is available for use by any process when determining an appropriate 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 the following options:
- The accounts are monitored by the Overdue Monitor (this is described in this chapter).
- The accounts are not monitored for overdue debt.

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 taxpayer 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 compliance rating
- Create a collection case to manage actions such as creating pay plans and referring debt to collection agencies
- … (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; for instance, send a letter, send a To Do entry to a user or write-off debt.

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-product 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 taxpayers before implementing the templates on the entire taxpayer 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 individual taxpayers starts with a "gentle reminder" letter followed 10 days later by a letter threatening to place a lien to secure the debt 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.

After setting up the above, the *Overdue Rule Plug-In* will use the challenger template X% of the time rather than the champion template.

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-product 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-product algorithm (*C1-OBASM-AMT*) if you collect on overdue assessments or obligations. The base algorithm uses the **Obligation Type - Determine Detailed Balance** algorithm to calculate the amounts.

Overdue Processes Are Highlighted Elsewhere

The topics in this section describe how other parts of the system highlight the existence of overdue processes.

Contents

Alert Zone

If you plug-in the appropriate alert algorithm (*C1-OD-PROC*) on the Installation record, alert(s) will be shown for active overdue processes in the Alert Zone that appears in the Dashboard and on Control Central - Account Information.

Compliance Zone

The Compliance zone on *Control Central - Account Information* shows **active** overdue processes.

Account Activity History Zone

The Account Activity History Zone on *Control Central - Account Information* shows **pending**and **waiting** events and **inactive** processes.

How Are Overdue Processes Cancelled?

A user may cancel an overdue process at their discretion, online / real-time using *Overdue Process - Main*.

The system will automatically cancel an overdue process when the object(s) associated with the overdue process are sufficiently paid. Exactly when the system checks if an overdue process should be cancelled is dependent on your organization's billing and accounting rules. For example, if you practice *open-item accounting*, you'd want to analyze an account's active overdue processes whenever a match event is added, changed or deleted (as match events are the only objects that impact if debt is considered paid in an open-item world). The base-product supports this specific example. Alternatively, if you plug in the base product Account Type - FT Freeze algorithm (*C1-CFTZ-CMRV*), an account's overdue processes will be reviewed for cancellation whenever a credit FT is frozen for the account. If you need additional events to check if an overdue process should be canceled, a base-product 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 that 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 debt. 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*.

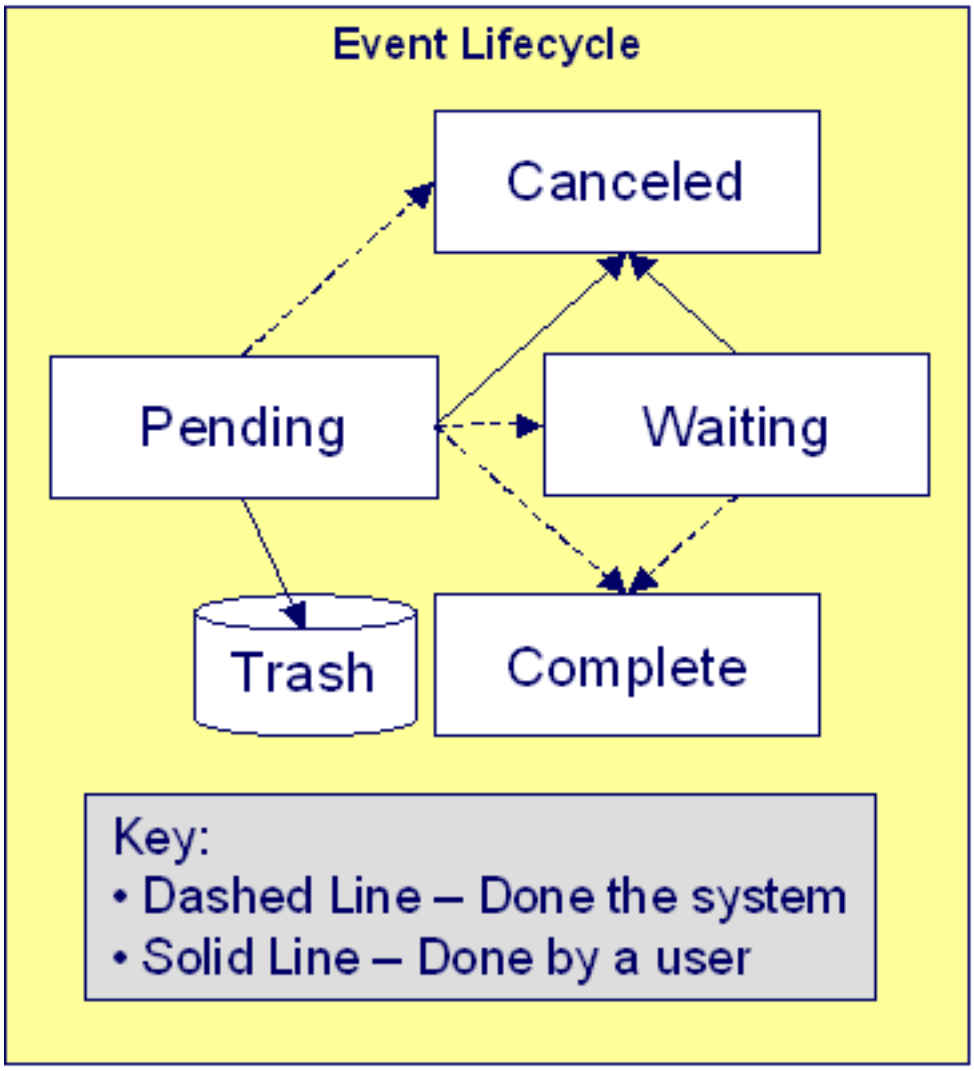
Overdue Event Information Is Overridable

- "Overdue event info" is the concatenated string of information that summarizes an overdue event throughout the system. The base-product logic constructs this string by concatenating the following information:
- The event type's description
- The event's status
- If it's **pending**:
- If the event has a trigger date, the number of days until it's triggered plus the verbiage **day(s) from today**
- Otherwise, the verbiage **dependent on other events**
- If it's **waiting**, the number of days, hours and minutes that it's been waiting
- If it's **canceled**, the cancel reason code's description
- If it's **complete**, the number of days, hours and minutes that it's been complete

If you'd prefer a different info string, you can develop a new algorithm and plug-it in on your event types. This design allows some / all event types to have an override info string.

Overdue Event Lifecycle

The following diagram shows the possible lifecycle of an overdue event:



Overdue events are initially created in the **Pending** state. An event can take myriad paths after it's created; it all depends on how you've configured the system. The following topics describe an event's lifecycle:

How and When Events Are Activated

An overdue event contains the date it should be activated; this is referred to as its trigger date. On this date, the Overdue 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 another process, such as an appeal by the taxpayer, is outstanding. Then, when the other process is complete, the overdue process can start up where it left off. Currently the base provides an algorithm type (*C1-HIOCE*) to suspend an overdue process while an account is linked to an open *case* of a given type.

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 product does not contain the algorithms you need, simply develop new ones using the base-product 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 product cancel algorithms insert a *log entry* when a user manually cancels an event.

The Big Picture of Collection Cases

In many tax authorities collecting unpaid debt is typically done in two phases: a series of unmonitored actions, typically letters, attempting to convince the taxpayer to pay, and a series of user-oriented actions such as contacting the taxpayer, setting up payment plans, and referring debt to a collection agency. The user-oriented actions are handled using a collection case. While not required, it is expected that many overdue processes will create a collection case when the overdue events did not prompt payment of the debt.

The topics in this section provide background information about collection cases.

Collection Case Overview

A collection case provides the functionality to record and track the interactions between the taxpayer and the user responsible for the collection. Many actions can take place once a collection case is created:

- A pay plan can be created
- The debt may be referred to a collection agency
- Letters may be sent to the taxpayer to advise of additional penalties
- A lien may be placed to secure the debt

How Are Collection Cases Created?

The activation of an overdue event creates a collection case. It is possible for an overdue process to be linked to more than one collection case over time but only one collection case can be active for an overdue process at a given time. The base package provides a sample overdue event activation algorithm (*C1-CR-COL-CS*) to create a standard collection case.

Collection cases are created for persons not accounts. It is possible for collection cases that are linked to different overdue processes to be consolidated into a single case.

It is important to note that a collection case exists with respect to one or more overdue processes. A collection case's overdue process defines the objects in arrears. These objects are monitored to determine if the overdue process can be cancelled and the collection case can be closed.

Collection Case Lifecycle

The lifecycle of the collection case depends upon the configuration of the associated business object. The base package includes a sample standard collection case with a simple lifecycle, as follows:

- The collection case is initially created in the **Pending Investigation** state.
- The case will transition to the **Actions In Progress** state the first time a user initiates an action, such as creating a pay plan or sending a letter. Since many collection case actions happen in parallel, the expectation is that the case

remains in this state until closed. Monitoring algorithms can be configured to check whether any of the actions in progress for the case have been waiting too long.
- The case may be **Transferred** to another collection case type or to an existing collection case.
- The user can manually transition the collection case to the **Uncollectible** state if they determine that no further action is to collect the debt is possible.

In addition to the standard actions for edit and state transition, a collection case may have special actions that apply to this type of collection, such as creating a pay plan or a collection agency referral. Additional actions are configured via the maintenance UI map. Refer to the base business objects for further details of the sample configuration.

How Are Collection Cases Closed?

Collection cases can be closed at the user's discretion or when the associated overdue process is cancelled. The base package provides a sample **Cancel Logic** algorithm (*C1-CL-COLLCS*) that closes any open collection cases linked to the process provided there are no other active overdue processes for the case.

Overdue Events Wait For The Collection Case To Conclude

Overdue events that create collection cases are perfect examples of events that *wait* for the object they create to complete before they, in turn, **complete.** After the collection case concludes, the originating overdue event will complete thus triggering its dependent events, such as writing off the debt. The base package provides a sample **Monitor Waiting Event** algorithm that checks whether all collection cases associated with the overdue process are in a final state.

Collection Case Type Defines Parameters

For each type of collection case, you must configure a collection case type to capture the appropriate parameters needed by the collection case actions. Typical parameters include:

- The To Do Type used to notify the responsible user that a new collection case has been created
- The default pay plan type to be used for a collection case of this type
- The To Do Types to be used to notify the responsible user if certain actions have been outstanding for too long without a response, such as a collection agency referral with no updates.

The Big Picture Of Collection Agency Referrals

Before debt is written off, many implementations refer the unpaid debt to a collection agency. The following topics describe how collection agency referrals are managed.

Collection Agency Referrals Overview

The system creates a *Collection Referral* record for a collection agency to track the debt that is to be collected. A collection referral is linked to an account. Collection referrals have history records that contain the amount of debt referred to the agency. Creating a history record 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.

How Are Collection Agency Referrals Created?

Users can create collection agency referral manually or by configuring an overdue event type with an activation algorithm that creates the referral. The base package sample activation algorithm (*C1-OE-AGYREF*) will refer the total unpaid debt for the overdue process to the collection agency with the least amount of referred debt. If you prefer different logic, you must develop your own algorithm.

In many tax authorities, creating a collection agency referral is one of the actions that would typically take place for a collection case. The base product BO for collection case **C1-StandardCollectioncase** can be used as an example of how to include the functionality to create a collection agency referral and link it to the collection case.

Cancelling Collection Agency Referrals

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). You can cancel a referral manually by simply creating a new collection agency referral history record (with a type of **cancel**).

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. If your implementation chooses to create collection agency referrals via overdue events, we strongly recommend plugging in an algorithm that will cancel the referrals when an overdue process is cancelled. If you choose to manage collection agency referrals via collection cases, the base package provides a sample business object status **Enter** algorithm (*C1-CCC-AGCYR*) to cancel any referrals linked to the collection case when it is closed.

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. If your implementation chooses to create collection agency referrals via overdue events, you set can set up your overdue process template to have an event that creates a collection agency cancellation X days after the referral. The base package provides a sample event activation algorithm (*C1-OE-AGYCAN*) to cancel all active collection agency referrals associated with the overdue process

Note:

Log entry. The base-product overdue event activation algorithms that make and cancel collection agency referrals insert rows in the overdue process's *log* to audit these events.

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.

Starting Write-Off Oriented Events

While the system provides overdue event activation algorithms that manage write-off oriented actions, such as referral to collection agency, tax authorities typically handle those actions as part of collection case activity.

Most overdue process templates will be configured to contain an event that writes-off the unpaid balance of the debt. This event should be configured to be dependent on the event that created the associated collection case. A **Monitor Waiting Event**algorithm can detect the collection case is closed and complete the waiting event. This allows the subsequent write off event to be triggered.

Small Amount Write-Downs

Many organizations will write-down a debt whose value is small early in an overdue process. The base package overdue event activation algorithm to write off assessments (*C1-WRITE-OFF*) includes a parameter to support this requirement. (For example, indicate that you should write down an overdue process's assessment 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.

Write Offs And Overdue Process Cancellation

If an overdue event writes off debt, the state of the process depends on your cancel criteria and where the overdue event is positioned in the overdue process. For example, if an overdue process has an overdue event that writes off small amounts of debt early in the process, a process whose debt meets the threshold criterion will be canceled when the event activates.

Contrast this to an overdue process where the last event writes off the debt. Because there are no other events to activate, the process will complete (i.e., it will not be canceled).

Calendar vs. Work Days

When you set up your overdue templates, you supply information that controls how the event's trigger date is calculated. You have two options:

- You can say that an event's trigger date can only be populated after earlier, dependent events are complete. For example, the 2 nd event is triggered 2 days after the 1 st event is 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 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.

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.

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-product 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.
- *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-product.
- *Messages* that are used to define the verbiage in the *log entries*. For example, if you use the base-product 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-product algorithms (this means you should not have to set up new messages).

  The only way to compile the complete list is to design the parameters for each overdue event type algorithm. Refer to *Overdue Event Type - Main* for the supported plug-in spots.

  After you've set up the objects referenced on the algorithms, you can then set up the algorithms. Only then can you set up the overdue event types.

Overdue Process Templates

After your overdue event types exist, you can set up your overdue process templates. You will find that most of the time spent setting up your overdue process templates is spent setting up the objects that are referenced on the overdue process template algorithms. Refer to *Overdue Process Template - Main* for the supported system events.

Collection Classes

When setting up *collection classes*, 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-product algorithm.

Collections - Installation Options

Control tables required to collect on overdue bills are described in *Overdue Processing - Setup Tasks*. For more information about collections installation options, refer to *Installation Options - Collections*.

Standard Collection Case Type

The base product supplies the business object **C1-StandardCollectionCase**, which is designed to cater for collection cases that incorporate standard actions such as creating pay plans and collection agency referrals.

The base product also supplies the business object **C1-StandardCollectionCaseType**, which defines the configuration information used by the standard collection case. To enable this functionality the following configuration tasks are needed:

- Define an appropriate To Do type and To Do role for sending a notification that a new collection case has been created for an overdue process.

- Define a Customer Contact class and type that reference the default letter template to be sent to a taxpayer for a collection case of this type.
- Define an appropriate To Do type and To Do role for sending a notification that the maximum days to wait after sending a letter before additional activity should occur has been exceeded.
- Define an appropriate To Do type and To Do role for sending a notification that the maximum days to wait after referring debt to a collection agency has been exceeded.
- Define a default Pay Plan obligation type for collection cases of this type
- Define appropriate reasons for closing a collection case due to the debt being uncollectible. Uncollectible reasons are defined using a customizable *lookup*. The lookup field name is **C1_COLL_CASE_UNCOLL_RSN_FLG**.
- Define a collection case type for the business object **C1-StandardCollectionCase**

Your implementation can define additional business objects to support collection case processing.

Feature Configuration

You must set up a *Feature Configuration* to define parameters that control various overdue processing options. The following is an example of how the Feature Configuration would look for an implementation:

The following points describe the various **Option Types** that must be defined:

- **Trigger Date Option** 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.
- **Champion Challenger Option.** 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 **ABnnn** 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 rule algorithm 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 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.

Alert To Highlight Active Collection Cases

If you want an alert to appear if the account has active collection cases, you must configure an appropriate **Control Central Alert** algorithm (*C1-CCAL-COCS*). 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

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 product 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 product algorithms as an example of how to do this.

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 and collection procedures.

Setting Up Overdue Event Types

An overdue event type encapsulates the business rules that govern a given type of overdue event. Open this page by selecting **Admin Menu, Overdue Event Type**.

Note:

Recommendation. Before using this transaction, we strongly recommend that you review *The Big Picture Of Overdue Events*.

Description of Page

Enter a unique **Overdue Event Type** code and **Description** for the overdue event type.

Use **Long Description** to provide a more detailed explanation of the purpose of the overdue event type.

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

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to **10** unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	*Optional / Required*	*Description*
Cancel Logic	Required	This algorithm is executed to cancel an overdue event. Refer to *How Are Events Canceled* for the details. Click *here* to see the algorithm types available for this system event.
Event Activation	Required	This algorithm is executed to activate an overdue event on its trigger date. Refer to *Overdue Events Can Do Many Things* and *How and When Events Are Activated* for the details. Click *here* to see the algorithm types available for this system event.
Event Information	Optional - only used if you want to override an overdue event's info string	This algorithm is executed to construct an overdue event's override info string. Refer to *Overdue Event Information Is Overridable* for the details. Click *here* to see the algorithm types available for this system event.
Monitor Waiting Events	Optional - only used if events of this type can enter the **Waiting** state	This algorithm is invoked by the Overdue 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 Overdue Process Templates

An overdue process template encapsulates the business rules that govern a given type of overdue process. Open this page by selecting **Admin, Overdue Process Template**.

Note:

Recommendation. Before using this transaction, we strongly recommend that you review *The Big Picture Of Overdue Processes*.

Description of Page

Enter a unique **Overdue Process Template** and **Description** for the overdue process template.

Collecting On Object defines the type of object managed by this overdue process. This field actually references a *foreign key characteristic type* that references the managed object. For example, if this overdue process template manages overdue bills, you'd reference a foreign key characteristic that references the bill object.

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

- Specify the algorithm's **System Event** (see the following table for a description of all possible events).
- Specify the **Algorithm** to be executed when the System Event executes. Set the **Sequence** to **10** unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event** (note, all system event's are optional and you can define an unlimited number of algorithms for each event).

System Event	***Optional / Required***	***Description***
Calculate Unpaid & Original Amount	Required	This algorithm is executed to calculate the unpaid and original amounts of the objects associated with the overdue process. These amounts are shown on the overdue process page and in the base-product *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 **Dep 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 **Dep 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 **Dep on Other Events** is on, define the events that must be completed or cancelled before the event will be triggered.

- **Sequence** is system-assigned and cannot be specified or changed.
- **Dependent on Sequence** is the sequence of the dependent event.

Where Used

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

Setting Up 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.

Description of Page

Enter a unique **Collection Class** code and **Description** for each collection class.

Indicate a **Collection Method** of **Overdue** if the accounts belonging to this collection class are subject to overdue collection procedures. If accounts belonging to this collection class are not subject collection, 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 Collection Class Overdue Rules

Collection class overdue rules contain algorithm that impact accounts associated with a given collection class, division and currency code are managed. Open this page by selecting **Admin, Collection Class Overdue Rules**.

Note:

Recommendation. Before using this transaction, we strongly recommend that you review *Different Overdue Rules For Different Taxpayers*.

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*
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.

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 this page by selecting **Admin, Overdue Event Cancel Reason**.

Description of Page

Enter an easily recognizable **Overdue Event Cancel Reason** and **Description** for each cancellation reason.

Where Used

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

Setting Up Collection Case Types

A **Collection Case Type** defines the configuration information that is common to collection cases of a given type. The type of information captured on the collection case type is governed by the collection case type's business object.

To set up a Collection Case Type, select **Admin Menu, Collection Case Type**.

The topics in this section describe the base-package zones that appear on the Collection Case Type portal.

Collection Case Type List

The Collection Case Type List zone lists every collection case type. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent collection case type.
- The standard actions of **Edit, Duplicate** and **Delete** are available for each collection case type.
- State transition buttons are available to transition the collection case type to an appropriate next state.
- Click the **Add** link in the zone's title bar to add a new collection case type.

Collection Case Type Actions

This is a standard actions zone. The Edit, Delete and Duplicate actions and appropriate state transition buttons are available.

Collection Case Type

The Collection Case Type zone contains display-only information about a Collection Case Type. This zone appears when a Collection Case Type has been broadcast from the Collection Case Type List zone or if this portal is opened via a drill down from another page.

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

Collection Case Type Log

This is a standard *log zone*.

Where Used

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

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 are assigned to collection agency referrals when the collection agency referral background process executes. Refer to *The Big Picture Of Collection Agency Referrals* for more information.

ConfigLab Addendum

This chapter is an addendum to the *general ConfigLab chapter*. This addendum describes ConfigLab functionality that is specific to Oracle Enterprise Taxation Management.

Account Staging

Some organizations periodically transfer account-related data (e.g., bills, payments, etc.) to a "test" environment so realistic data can be used to test new configuration data. If your organization does this, you can use this transaction to define the "stable" of accounts to be periodically transferred. Use **Admin, Account Staging** to access this transaction.

Description of Page

This page is dedicated to a grid that shows the accounts to be copied to an environment by a compare DB process. These accounts are defined in respect of a given Environment Reference (meaning you can have a different set of accounts for different target environments). The following information appears in the grid.

- **Account ID** identifies a given account.
- **User** defines the user who added the account.
- **Create Date/Time** is the date and time the account was added to account staging.

Note:

Update the DB process that manages the transfer. After defining these accounts, you should update the DB process that transfers accounts to have a primary instruction that references the **ACCT STG** maintenance object. You do this in the environment to which the above accounts will be transferred (as this is the environment that you'll submit the comparison process in).

Oracle Policy Automation Integration

Oracle Enterprise Taxation Management provides tools to facilitate the integration with Oracle Policy Automation's determinations services (OPA). The following sections describe the technical information needed by your implementers to be able to invoke OPA rules from within the system.

Configuring the Integration

The topics in this section describe configuration required in the product for integration with OPA and steps needed to use one of the sample OPA Rulebases provided by the product.

Verify the Rulebase Web Service Adapter Options

The integration with OPA uses *web service adapters* to define the connection information to the OPA server and the specific mapping required between the product and the individual OPA rulebases. The product provides a Rulebase Web Service Adapter BO (**F1-OpaWebSvc**). Each OPA Rulebase requires an individual web service adapter that uses this business object.

Before creating any web service adapter records, verify that the Rulebase Web Service Adapter BO has been properly configured for the installation. Each installation must configure two BO options with specific server locations related to the integration. Review the BO options to ensure they are configured. If not, the installation team can provide the appropriate values:

- **Oracle Web Determinations Server Location**. This is the URL of the Oracle Web Determinations server. It should be in the format **http://[server-name]:[portNumber]/determinations-server/soap**. For example **http://opa-server-name:1000/determinations-server/soap**. This is the server location that is part of any WSDL URL used to create an appropriate web service adapter for a specific rulebase.
- **Oracle Policy Automation Rulebase Location**. This is the server directory that includes a copy of the deployment files of the OPA rulebases supplied with the product. This is needed by the business service that generates the mapping data areas when creating a web service adapter for a rulebase.

Once the configuration of the Rulebase Web Service Adapter BO is verified, your implementers may now create specific web service adapters for each specific rulebase used by your implementation.

Configuration Required for a Sample OPA Rulebase

For any sample OPA rulebase provided by the product, most components are supplied by the product, but final steps are required by each implementation. To illustrate the details, we'll use as an example the timeliness validation algorithm for creating an appeal.

- The product supplies the rulebase. In this scenario it is called **Appeal_Validation**. As part of the installation of the product, the deployment files are installed on the OPA server that the product will integrate with and also copied to the rulebase location server directory described above.
- The product supplies an appropriate algorithm type that invokes the call to OPA. The algorithm type is responsible for gathering the data required as input to the OPA rulebase and includes appropriate logic based on the outcome returned by the rulebase. For example, the call to OPA may return a decision report. The algorithm may be designed to store the decision report and link the decision report ID to the appropriate entity for viewing later. In this scenario the product supplies a BO Enter plug-in for the Appeal business object's **Validated** state. It is called *C1-AP-CHKATL* . The product supplies the algorithm type. However, the algorithm requires the Web Service

Adapter name as input so the product does not supply the algorithm nor is it plugged into the appeal business object.

The following points explain the additional configuration steps required by an implementer to implement a sample rulebase.

1. Determine the WSDL URL for the rulebase. The WSDL URL is composed as follows **[Oracle Web Determinations Server Location]/[rulebase name]/specific?wsdl**. Using the URL example above and this rulebase the WSDL URL would be **http://opa-server-name:1000/determinations-server/soap/Appeal_Validation/specific?wsdl**.

   Tip:

   To verify that you have composed the WSDL URL correctly, paste the composed URL into an internet browser and verify that you can access the WSDL.
2. Define a web service adapter for the rulebase using the rulebase WSDL. Refer to *Understanding Web Service Adapters* for details about fully creating and generating the web service adapter.
3. Create an algorithm for the appropriate algorithm type provided by the product for this particular OPA integration. The algorithm requires the web service adapter name. Depending on the specific algorithm type, there may be additional parameters required.
4. Plug the algorithm into the appropriate plug-in spot as defined by the specific integration point.

Fastpath:

For this scenario, refer to *Configuring the Appeal Timeliness Rule* for specifics about defining the algorithm correctly and plugging it into the appropriate plug-in spot.

Defining a New Rule

The first step is to design the business rule. In addition to designing the logic of the OPA rulebase, your designers must consider where and when the business rule should be invoked by the product as well as what logic should occur in the product based on the output from the call to OPA.

The topics in this section describe more detail to consider when implementing a new business rule using an OPA rulebase.

Develop the Algorithm to Invoke the Rule

An important step in considering an integration point with OPA is to consider when it should be invoked within the product. In other words, what plug-in spot? For example:

- Is the business rule validation that must be invoked any time a certain object changes? Then perhaps this should be implemented as a BO validation algorithm for an appropriate business object.
- Is the business rule validation that determines if an object may transition to a different status? Then perhaps this should be implemented as an enter plug-in algorithm on an appropriate state for a business object.

As part of determining the appropriate plug-in spot, the designer should become familiar with the expected responsibilities of the plug-in spot and the data it receives when invoked.

Next, the designer should determine if the product already supplies an algorithm type for the desired plug-in spot that invokes OPA. For example, perhaps the product supplies an out of the box integration with OPA and the algorithm logic (the data it retrieves to send to OPA and the logic that occurs after returning from OPA) is appropriate for your business rule. However, the detailed logic performed by the product's sample rulebase does not match the business rule needed by the implementation. If so, your implementation may only require a new OPA rulebase to plug into this algorithm type.

The following points highlight more detail needed for designing a new algorithm to invoke an OPA rulebase:

- Work with the business rule designer to determine what information needs to be provided to OPA. The algorithm is responsible for gathering all the information required by OPA, if it is not already supplied to the algorithm.

- To invoke OPA, the product provides a web service dispatcher business service (**F1-InvokeWebService**). The business service requires the web service adapter name, the name of the operation to invoke (typically **Assess** for OPA integration) and the request and response data areas related to the operation. The algorithm must determine the appropriate web service adapter and must populate the request data area with the information required for the OPA call.

  The product algorithms have typically been designed to define the web service adapter name as an input parameter. The algorithm then includes logic to retrieve the request and response data areas defined on the web service adapter. This allows for algorithm type re-use. It means that the same algorithm type may be used for different business rules implemented with different OPA rulebases, assuming that the other logic performed by the algorithm applies to the different rules.
- Determine what logic should occur upon receiving the rulebase response. See the Rulebase Outcome section below for more information.

Rulebase Outcome

The type of data returned by a rulebase is configured in the request by populating the Outcome Style element for each output value (referred to as a "goal" in OPA terminology).

- An outcome style of **value-only** tells OPA to simply return the value of the goal (output) . This is the default if nothing is specified in the request.
- An outcome style of **decision-report** tells OPA to return a detailed report of why a particular result is given in addition to the value of the goal.

> **Note:**
>
> The above values are defined in the Outcome Style extendable lookup (**F1-OutcomeStyleLookup**).

Based on the business requirements, the algorithm must determine what should occur based on the results.

If the business requirement dictates that the decision report should be stored, the product provides a business service Create Decision Report (**C1-CreateDecisionReport**). This business service stores the decision report details in a special "attachment" table and returns the ID. Based on the business rules, the ID may be captured and stored with the object that requested the call to OPA. The following points provide more detail on storing a decision report.

- The following information should be provided to the business service to create the decision report:
- Populate the BO with **F1-DecRpt**
- Populate the MO and prime key values to appropriate values based on the object that invoked the OPA rule. For example, if the OPA rulebase was used to validate an appeal, the MO is set to **C1-APPEAL** and the PK value is set to the appeal id. This allows cross referencing on the decision report to the object that created it.
- Populate the Decision Report Data with the appropriate decision report node returned by the call to OPA.
- The attachment id is returned. To store a link to the decision with the object that created it, the product recommendation is to create a Log Entry, if the maintenance object supports logs. The characteristic type for the log should be set to **C1-DCRPT** and the characteristic value should be set to the attachment ID.

> **Note:**
>
> The characteristic type may require additional configuration to include the log in question as one of the valid characteristic entities.

Refer to *Viewing Decision Report* for information about viewing the decision report.

Deploying the Rulebase

Use Oracle Policy Modeling to develop the rulebase. Once the rulebase is tested and it is compiled and built, the deployment file is created.

- This rulebase should be deployed on the Oracle Web Determinations server. Note that in most implementations a separate group such as a release services group is responsible for this type of task.

- As described in *Configuring the Integration*, besides deploying the rulebase on the server, the web service adapter logic requires the rulebase to be copied to the server directory defined in the BO option Oracle Policy Automation Rulebase Location.

Final Steps

The remaining steps for the integration involve creating the web service adapter and fully configuring and plugging in the algorithm.

➤ **Fastpath:**

Refer to *Configuring the Integration* for more information.

Viewing the Decision Report

The product provides examples of storing a resulting decision report after a call to an OPA rulebase. The appeal - check timeliness algorithm stores the resulting decision report and provides a link to the decision report via the appeal log.

➤ **Fastpath:**

Refer to *C1-AP-CHKATL* for more information.

As described in *Rulebase Outcome*, a stored decision report can be linked to the calling object using a log entry and its characteristics. The section indicates an appropriate foreign key characteristic type to use.

A user views the decision report by navigating to the log and clicking the hyperlink for the stored decision report (attachment) id. Assuming that the appropriate characteristics type was used, clicking the hyperlink causes the decision report to display in a separate window.

CTI-IVR Integration

Oracle Enterprise Taxation Management 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 Enterprise Taxation Management
- 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 Enterprise Taxation Management 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 Enterprise Taxation Management 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 Enterprise Taxation Management 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 Enterprise Taxation Management.
- 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 Enterprise Taxation Management 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 Enterprise Taxation Management.

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 taxpayer from within the system, a context menu item **Go To Automated Dialer** is available on the Person context menu. To call a taxpayer 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

Micorsoft Phone Dialer Configuration

If your implementation chooses to use the functionality provided with the system and integrate with Microsoft Phone Dialer, you must copy the JSP page ext_cti_dialer.jsp from the /cm_templates directory found under the web application root directory on your Oracle Enterprise Taxation Management 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 Enterprise Taxation Management 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, 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 BPA script **C1-GetNxtClr** is provided and can be used to request the next call waiting in an inbound queue managed by a CTI application. Your implementation may choose to configure a menu entry for this BPA script or may recommend that appropriate users configure this BPA script as a "favorite" script and enable the Favorite Scripts dashboard zone.

When the BPA is invoked, 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 Enterprise Taxation Management 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");
```

The Conversion Process

This document describes the Oracle Enterprise Taxation Management conversion process.

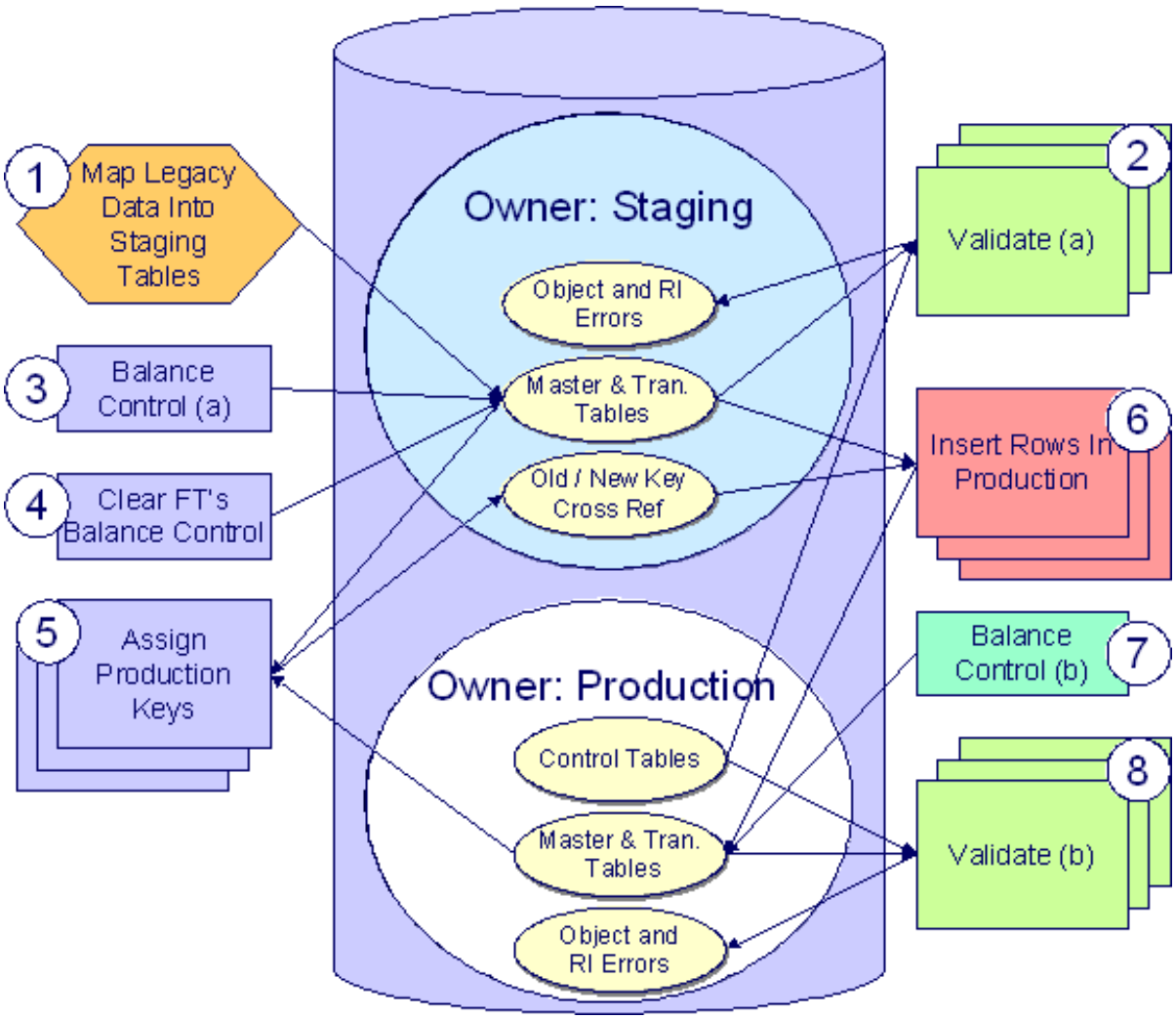
Introduction

When you're ready to convert data from your legacy system into Oracle Enterprise Taxation Management, you will have analyzed your requirements according to your business and organizational needs and set up the control tables accordingly.

Fastpath:

Refer to the **Administration Guide** for a complete discussion of the various control tables and the order in which they must be set up.

After the control tables are set up, you are ready to load data into the system from your legacy system. This conversion effort involves several steps as illustrated in the following diagram:



The following points briefly outline each of the above tasks:

- **Map Legacy Data Into Staging.** During this step, your legacy master data (e.g., account, person, location) and transaction data (e.g., bills, payments) is migrated into the system. Notice that you are not migrating this data directly into production. Rather, your rows are loaded into tables that are identical to the production tables; they just have a different owner. Refer to *Appendix B - Multiple Owners In A Single Database* for information about table ownership.

⚠ **Caution:**

Different Databases For Staging and Production. The above diagram illustrates how the system is configured to support the conversion effort in the standard installation, i.e., the staging tables are in the same database as the production tables (each with a different owner). However, it is possible for the staging tables to be in a separate database. This option requires additional effort on your part (because you would have to copy the control tables from production into your staging database). Please refer to *Appendix B - Multiple Owners In A Single Database* for information about this alternative.

Mapping legacy data into the system is probably the most challenging part of the conversion process because the system is a normalized database (and most legacy applications are not).

- **Validate (a).** During the validation (a) step, the system validates the data you loaded into the staging tables. Two types of validation programs exist:
- **Object Validation Programs.** Each of the system's master data objects (e.g., person, account, location, etc.) is validated using the same logic that is used to validate data added by users in your production system.

- **Referential Integrity Validation Programs.** After you have successfully validated the master data objects, the referential integrity validation programs are executed to validate transaction data and to highlight "orphaned" rows. These programs check the validity of the foreign keys on all rows on all tables.

Note:

Control tables from production. It's important to notice that the validation programs validate your staging data using the control tables that have been set up in production. Refer to *Appendix B - Multiple Owners In A Single Database* for a description of how this works.

- **Balance Control (a).** During this step, you run the balance control program and then verify that the balances that it generates are consistent with the balances in your legacy system.
- **Clear FT's Balance Control.** In the previous step, the system creates a balance control and links it to the FT's. If the balance control's balances are consistent with the amount of receivables being transferred into the system, you should run the Clear FT's Balance Control program. This program simply resets the Balance Control column on the FT so that the FT's can be included in a balance control (see the last step below) after they have been transferred to production.
- **Assign Production Keys.** During this step, the system allocates random, clustered keys to the rows in the staging database.
- **Insert Rows Into Production.** During this step, the system populates your production tables with rows from the staging. When the rows are inserted, their prime keys are reassigned using the data populated in the previous step.
- **Balance Control (b).** During this step, you run the balance control program against production. You do this to verify the balances in production are consistent with the values of receivables converted from your legacy application.
- **Validate (b).** During this step, you rerun the object validation programs, but this time against production. We recommend rerunning these programs to confirm that the insertion programs have executed successfully. We recommend running these programs in random sample mode (e.g., validate every 1000th object) rather than conducting a full validation in order to save time. However, if you have time, you should run these programs in full validation mode (to validate every object).

Conversion Process Steps

The following sections provide more details about the steps in the conversion process.

Map Legacy Data Into Staging Tables

This section provides some high level discussion about mapping legacy data to the system's staging tables. Refer to *The Staging Tables* for details about the staging tables in the system.

Note:

Recommendation. You can use any method you prefer to load Oracle Enterprise Taxation Management data from your legacy application. However, we recommend that you investigate your database's mass load utility (as opposed to using insert statements) as the mechanism to load the staging tables. In addition, we strongly recommend that you disable the indexes on these tables before populating these tables and then enable the indexes after populating these tables.

A Note About Keys

The prime keys of the tables in the staging database are either system-assigned random numbers or they aren't. Those tables that don't have system-assigned random numbers have keys that are a concatenation of the parent's prime-key plus one or more additional fields.

Every table whose prime key is a system-assigned random number has a related table that manages its keys; we refer to these secondary tables as "key tables". The following points provide more information about the key tables:

- Key tables are used by programs that allocate new keys. For example, before a new account ID is allocated, the key assignment program checks the account key table to see if it exists.
- Key tables exist to support archiving and ConfigLab requirements.
- When an object is *archived*, its row is removed from the primary table, but its key remains on the key table. This is done so that they key isn't reused in production, as we want to be able to reinstate an archived object.
- From a *ConfigLab* perspective, these key tables are used to prevent a key from being reused in production for an object added in a ConfigLab. For example, a user might add an account in a ConfigLab environment and we don't want its key to be allocated to an account added in production.
- Key tables only have two columns:

  1) The key of the object and 2) An environment ID. The environment ID identifies the database in which the object resides.
- Key tables are named the same as their primary table with a suffix of "_K". For example the key table for CI_ACCT is CI_ACCT_K.

  The name of every table's key table is defined under the Generated Keys column in the Table Names sections in *The Staging Tables*
- When you populate rows in tables with system-assigned keys, you must also populate a row in the related key table. For example, if you insert a row into CI_ACCT, you must also insert a row into CI_ACCT_K. The environment ID of these rows must be the same as the environment ID on this database's *installation record*.
- When you populate rows in tables that reference this record as a foreign key, you must use the appropriate key to ensure the proper data relationships. For example, if you insert a row in CI_SA for the above account, the ACCT_ID column must contain the temporary account key.
- When you insert rows into your staging database, the keys do not have to be random, system-assigned numbers. They just have to be unique number. A later process, *Allocate Production Keys*, will allocate random, system-assigned keys prior to production being populated.

Validate Information In The Staging Tables

During the first validation step, the system validates the data you loaded into the staging tables. Two types of validation programs exist:

- **Object Validation Programs.** The object validation programs validate each of the system's master data objects (e.g., person, account, location, etc.) and a limited number of transaction data objects (e.g., billable charge). Please note that these programs call the same programs that are used to validate data added by users in your production system.
- **Referential Integrity Validation Programs.** After the master data objects have been validated, the referential integrity validation programs are executed to validate transaction data and to highlight "orphaned" rows. These programs simply check the validity of the foreign keys on all rows on all tables.

The contents of this section describe how to execute the validation programs.

Object Validation Programs

Each of the objects described under *Master Data* must be validated using the respective object validation program indicated in its Table Names section.

In a limited number of cases object validation is available for *Transaction Data* objects, where customers may convert transaction data that is still pending. For the same objects you may also be converting historic records. You may not want to perform validation on completed records. As a result the background processes provided for transaction data allow you to limit the validation to records in a given status.

We strongly recommend validating each object in the following steps:

- Execute each object's validation program in random-sample mode to highlight pervasive errors. When you execute a validation in random-sample mode, you are actually telling it to validate every X records (where X is a

parameter that you supply to the job). Refer to *Submitting Object Validation Programs* for more information about the parameters supplied to these background processes.

- You can view errors highlighted by validation programs using the *Validation Error Summary* transaction.
- Correct the errors using SQL. Note, you can use the base package's transactions to correct an error if the error isn't so egregious that it prevents the object from being displayed on the browser.
- After all pervasive errors have been corrected; re-execute each object's validation program in all-instances mode to highlight elusive, one-off errors. Refer to *Submitting Object Validation Programs* for more information about the parameters supplied to these background processes.

Caution:

Take note! Whenever an object validation program is run, it is necessary to delete all previously recorded errors associated with its tables from the validation error table before it inserts new errors.

After the various object validation programs run cleanly, run the referential integrity validation programs as described in the next section.

Submitting Object Validation Programs

The object validation programs that are described in the *staging tables* table names matrices are classic background processes as they can also be run against production data. You submit these processes in the same way you submit any background process in production. Refer to *Object Validation Processes* for information about these processes and their parameters.

Referential Integrity Validation Programs

It's important to understand that only master data objects (e.g., persons, accounts, assets, locations, etc.) are validated by the object validation programs discussed above. This means that only master data objects will have their foreign keys checked for validity by the object validation programs. You must run the referential integrity programs to validate all other data.

The referential integrity validation programs highlight:

- Orphaned rows because orphan rows, by definition, don't reference an object.
- Invalid foreign keys on transaction data.

Note:

Validating Transaction Data. You may wonder why transaction data is not subject to the object validation routines. This is because: a) the production system only needs validation logic for master data because transaction data is not entered by users, and b) most conversions necessitate loading skeletal transaction data because the legacy system typically doesn't contain enough information to create accurate transactions in the system.

Each of the tables described under *Transaction Data* must be validated using the respective referential integrity validation program indicated in its Table Names section. We strongly recommend validating each table in the following steps:

- Execute each table's referential integrity validation program. Refer to *Submitting Referential Integrity Validation Programs* for more information about the parameters supplied to these background processes.
- You can view errors highlighted by this process using the *FK Validation Summary* transaction.
- Correct the errors using SQL (you cannot use the application to correct these types of errors).
- Rerun the referential integrity programs until no errors are produced.

Caution:

Whenever you run a referential integrity validation program, it deletes all errors associated with its table from the referential integrity error table.

In order to highlight orphaned rows in the master data, run the referential integrity validation programs against all tables described under *Master Data* using the procedure described above.

Submitting Referential Integrity Validation Programs

The referential integrity validation programs described under *Master Data* and *Transaction Data* (in the Table Names matrices) are submitted using a batch driver program, **CIPVRNVB**, and this program is executed in the staging database. Please note that the referential integrity validation programs may also be run in the production environment on occasion, to determine the integrity of data in the production database.

Refer to *Referential Integrity Validation Processes* for information about these processes and their parameters.

You should supply the following parameters to this program:

- **Batch code.** The batch code associated with the appropriate table's referential integrity validation program. Refer to each table listed under *Master Data* and *Transaction Data* (in the Table Names matrices) for each referential integrity batch code / program.
- **Batch thread number.** Thread number is not used and should be left blank.
- **Batch thread count.** Thread count is not used and should be left blank.
- **Batch rerun number.** Rerun number is not used and should be left blank.
- **Batch business date.** Business date is the date supplied to the referential integrity validation programs and the date under which statistics will be logged.
- **Total number of commits.** Total number of commits is not used and should be left blank.
- **Maximum minutes between cursor re-initiation**. Maximum minutes between cursor re-initiation is not used and should be left blank.
- **User ID.** User ID is only used to log statistics for the execution of the batch job.
- **Password.** Password is not used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start** (Y/N), **trace program exit** (Y/N), **trace SQL** (Y/N) and **output trace** (Y/N). These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed.

Recommendations To Speed Up Validation Programs

The following points describe ways to accelerate the execution of the validation programs:

- Ensure that statistics are recalculated after data has been inserted into the staging tables. For Oracle database users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.
- *Object validation programs* should be run multi threaded.
- Execute shorter running validation processes (e.g., less records) first so that the error data can be analyzed while other processes are busy running.
- *Referential integrity validation programs* run fairly quickly without much tuning. However, additional benefits are gained by running several programs at the same time.
- Remember that the *object validation programs* can be run in "validate every n^{th} row". We recommend running these programs using a relatively large value for this parameter until the pervasive problems have been rectified.

Balance Control (a)

During this step, you run the balance control programs and then verify that the balances that it generates are consistent with the balances in your legacy system.

Note:

Submitting this process. You submit this process in the staging database. Refer to *The Big Picture of Balance Control* for more information about the balance control processes. Refer to *Balance Control* for information about the page used to view the balances generated by this process.

Clear FT Balance Control

In the previous step, the system created a balance control and links it to the FT's. If the balance control's balances are consistent with the amount of receivables being transferred into the system, you should run the Clear FT's Balance Control program. This program simply resets the Balance Control column on the FT so that the FT's can be included in a balance control (see the last step below) after they have been transferred to production. Note: the batch control ID of **CNV-BCG** is used to request this process.

Note:

Submitting this process. You submit this process in the staging database. Refer to *Reset Balances* for a description of this process and its parameters.

Allocate Production Keys

The topics in this section describe the background processes used to assign production keys to the staging data.

The Big Picture of Key Assignment

It's important to understand that the system does not overwrite the prime-keys or foreign keys on the rows in the staging database, as this is a very expensive IO transaction. Rather, a series of tables exist that hold each row's old key and the new key that will be assigned to it when the row is *transferred into the production database*. We refer to these tables as the "old key / new key" tables. The old key / new key tables are named the same as their primary table, but rather than being prefixed by "CI", they are prefixed by "CK". For example, the old key / new key table for CI_ACCT is called CK_ACCT.

The insertion programs that transfer the rows into the production database use the new key for the main record of the key along with any other record where this key is a foreign key. Note that the capability of assigning the new key to a foreign key applies to

- "True" foreign keys, i.e. where the key is a column in another table. For example, CI_SA has a column for ACCT_ID.
- FK reference characteristics. These are characteristics that define, through an FK reference, the table and the key that this characteristic represents.

The insertion programs are not able to assign "new keys" to foreign keys defined in an XML structure field (CLOB).

The key assignment programs listed under *Master Data* and *Transaction Data* (in the table names sections) are responsible for populating the old key / new key tables (i.e., you don't have to populate these tables). Because the population of the rows in these tables is IO intensive, we have supplied detailed instructions that will accelerate the execution time of these programs.

Note:

Why are keys reassigned? The conversion process allocates new prime keys to take advantage of the system's parallel processing and data-clustering techniques in the production system (these techniques are dependent on randomly assigned, clustered keys).

Note:

Iterative conversions. Rather than perform a "big bang" conversion (one where all customers are populated at once), some implementations have the opportunity to go live on subsets of their customer base. If this describes your implementation, please be aware that the system takes into account the existing prime keys in

the production database before it allocates a new key value. This means when you convert the next subset of customers, you can be assured of getting clean keys.

Note:

Optional Foreign Keys. One responsibility of the key allocation programs is to generate a single "zero" entry in its key table to represent a blank foreign key. This step is required by the subsequent Insert programs so that they may correctly insert new rows where optional foreign keys are blank. For example, the CI_FT table has a foreign key to BILL_ID. If your implementation does not plan to convert bills but does plan to convert FTs, the key generation program for Bills must still be run in order to generate the entry in the CK_BILL table for the "zero" row. This is needed by the FT Insertion program. The details of which Key Generation programs are required for successful run of a given insert program are detailed in each staging table section.

Note:

Program Dependencies. The programs used to assign production keys are listed in the Table Names matrices. Most of these programs have no dependencies (i.e., they can be executed in any order you please). The exceptions to this statement are noted in *Program Dependencies*.

Submitting Key Assignment Programs

The key assignment programs described under *Master Data* and *Transaction Data* (in the Table Names matrices) are submitted using a batch driver program, **CIPVRNKB**, and this program is executed in the staging database. You should supply the following parameters to this program:

- **Batch code.** The batch code associated with the appropriate table's key assignment program. Refer to each table listed under *Master Data* and *Transaction Data* (in the Table Names matrices) for each key assignment batch code / program.
- **Batch thread number.** Thread number is not used and should be left blank.
- **Batch thread count.** Thread count is not used and should be left blank.
- **Batch rerun number.** Rerun number is not used and should be left blank.
- **Batch business date.** Business date is the date supplied to the key assignment programs and the date under which statistics will be logged.
- **Total number of commits.** Total number of commits is not used and should be left blank.
- **Maximum minutes between cursor re-initiation**. Maximum minutes between cursor re-initiation is not used and should be left blank.
- **User ID.** User ID is only used to log statistics for the execution of the batch job.
- **Password.** Password is not used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start** (Y/N), **trace program exit** (Y/N), **trace SQL** (Y/N) and **output trace** (Y/N). These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed.
- **Mode.** The proper use of this parameter will greatly speed up the key assignment step as described under *Recommendations To Speed Up Key Generation*. This parameter has three values:

- If you supply a mode with a value of **I** (initial key generation), the system allocates new keys to the rows in the staging tables (i.e., it populate the respective old key / new key table).
- If you supply a mode with a value of **D** (resolve duplicate keys), the system reassigns keys that are duplicates.
- If you supply a mode with a value of **B** (both generate keys and resolve duplicates), the system performs both of the above steps. This is the default value if this parameter is not supplied.
- Please see *Recommendations To Speed Up Key Generation* for how to use this parameter to speed up the execution of these processes.

Note:

Parallel Key Generation. Note well, no key generation program should be run (either in mode**I** or **B**) while another program is being run unless that program is in the same tier (see *Program Dependencies* for a description of the tiers).

- **Start Row Number.** This parameter is only used if you are performing conversions where data already exists in the tables in the production database (subsequent conversions). In an Oracle database the key assignment routines create the initial values of keys by manipulation of the Oracle row number, starting from 1. After any conversion run, a subsequent conversion run will start with that row number again at 1, and the possibility of duplicate keys being assigned will be higher. The purpose of this parameter is to increase the value of row number by the given value, and minimize the chance of duplicate key assignment.

Recommendations To Speed Up Key Generation Programs

The following points describe ways to accelerate the execution of the key generation programs.

Note:

Naming convention. The convention "CK_<table_name>" is used to denote the old key / new key tables described under *The Big Picture of Key Assignment*.

- Make the size of your rollback segments large. The exact size is dependent on the number of rows involved in your conversion. Our research has shown that processing 7 million rows generates roughly 3GB of rollback information.
- Setup the rollback segment(s) to about 10GB with auto extend to a maximum size of 20GB to determine the high water mark
- A next extent value on the order of 100M should be used.
- Make sure to turn off all small rollback segments (otherwise Oracle will use them rather than the large rollback segments described above).
- After the key assignment programs execute, you can reclaim the space by:
- Keep a low value for the "minimum extent" parameter for the rollback.
- Shrink the rollback segments and the underlying data files at the end of the large batch jobs.
- Compute statistics on the CK_<table_name> tables after every 50% increase in table size. Key generation is performed in tiers or steps because of the inheritance dependency between some tables and their keys. Although key generation for the tier currently being processed is performed by means of set-based SQL, computation of statistics between tiers will allow the database to compute the optimum access path to the keys being inherited from the **previous** tier's generation run. For Oracle database users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.
- Optimal use of the **Mode** parameter under *Submitting Key Assignment Programs*.
- Before any key assignments, alter both the "old key" CX_ID index and the "new key" CI_ID index on the CK_<table_name> tables to be unusable.
- Run all *key assignment tiers*, submitting each job with MODE = "I".
- Rebuild the CX_ID and CI_ID indexes on the CK_<table_name>. Rebuilding the indexes using both the PARALLEL and NOLOGGING parameters will speed the index creation process in an Oracle database. Statistics should be computed for these indexes.
- Run all key assignment tiers that were previously run in MODE = 'I', submitting each job with MODE = 'D'. This will reassign all duplicate keys.

Insert Production Data

The topics in this section describe the background processes used to populate the production database with the information in the staging database.

The Big Picture Of Insertion Programs

All insertion programs are independent and may run concurrently. Also note, all insertion programs can be run in many parallel threads as described in the next section (in order to speed execution).

Submitting Insertion Programs

The insertion programs described under *Master Data* and *Transaction Data* (in the Table Names matrices) are submitted using a batch driver program, **CIPVRNIB**, and this program is executed in the staging database. You should supply the following parameters to this program:

- **Batch code.** The batch code associated with the appropriate table's insertion program. Refer to each table listed under *Master Data* and *Transaction Data* (in the Table Names matrices) for each insertion batch code / program.
- **Batch thread number.** Thread number contains the relative thread number of the process. For example, if you want to insert accounts into production in 20 parallel threads, each of the 20 execution instances receives its relative thread number (1 through 20). Refer to *Parallel Background Processes* for more information.
- **Batch thread count.** Thread count contains the total number of parallel threads that have been scheduled. For example, if the account insertion process has been set up to run in 20 parallel threads, each of the 20 execution instances receives a thread count of 20. Refer to *Parallel Background Processes* for more information.
- **Batch rerun number.** Rerun number is not used and should be left blank.
- **Batch business date.** Business date is the date supplied to the insertion programs and the date under which statistics will be logged.
- **Total number of commits.** This is the number of commits IN TOTAL that you want to perform. For example, if you have 1,000,000 accounts and you supply a value of **100**; then a commit will be executed for approximately every 10,000 accounts.
- **Maximum minutes between cursor re-initiation**. This should only be populated if you want to override the default value of **15**.
- **User ID.** User ID is only used to log statistics for the execution of the batch job.
- **Password.** Password is not used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start** (Y/N), **trace program exit** (Y/N), **trace SQL** (Y/N) and **output trace** (Y/N). These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed.

Recommendations To Speed Up Insertion Programs

The following points describe ways to accelerate the execution of the insertion programs:

- Before running the first insertion program:
- Rebuild the index on the prime key on the old key / new key table (i.e., those tables prefixed with "CK").
- Re-analyze the statistics on the old key / new key table (i.e., those tables prefixed with "CK"). For Oracle database users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.
- Alter all indexes on the production tables being inserted into to be unusable.
- After the insertion programs have populated production data, rebuild the indexes and compute statistics for these tables. For Oracle database users, we strongly recommend using the Oracle-provided PL/SQL package to generate statistics rather than the analyze command.

Run Balance Control Against Production

During this step, you rerun the balance control program, but this time against production. You do this to verify the balances in production are consistent with the values of receivables converted from your legacy application.

Note:

Submitting this process. You submit this process in the production database. Refer to *The Big Picture of Balance Control* for more information about the balance control processes. Refer to *Balance Control* for information about the page used to view the balances generated by this process.

Validate Production

During this step, you rerun the *object validation programs*, but this time in production. We recommend rerunning these programs to confirm that the insertion programs have executed successfully. We recommend running these programs in random sample mode (e.g., validate every 1000 th object) rather than conducting a full validation in order to save time. However, if you have time, you should run these programs in full validation mode (to validate every object). Please refer to the various "Table Names" sections above for the respective names of the programs to run.

The Validation User Interface

The topics in this section describe the various pages that assist in the conversion effort.

Validation Error Summary

Navigate to the **Admin** menu and open **Validation Error Summary** to view validation errors associated with the objects defined in *Master Data*.

Description of Page

You can use**Table Name** to restrict errors to a specific object. If this field is left blank, all errors on all objects will be displayed.

The grid contains a separate row for each type of error. The following information is displayed:

- **Table Name** is the name of the main table associated with the object.
- **Message Category** and **Message Number** define the type of error. These fields are the unique identifier of the message that describes the error (the verbiage of this message is displayed in the **Message Text** column).
- **Count** contains the number of records with this error. Press the Go To button to see the individual records with the error.

Validation Error Detail

This page is used to view validation errors of a given type associated with one of the objects defined in *Master Data*. This transaction is not intended to be invoked from the **Admin** menu. Rather, drill into the validation details from *Validation Error Summary*.

Description of Page

Use **Table Name**, **Message Category**, and **Message Number** to define the object and the type of error you wish to display. The grid contains a separate row for each object with the given type of error. The following information is displayed:

- **Table Name** is the name of the main table associated with the object.
- **Record Identifier** is the unique identifier of the object with the error (e.g., the person ID, the account ID, the location ID, etc.). Press the Go To button to transfer to the maintenance page associated with the object.
- **Message Category** and **Message Number** define the type of error. These fields are the unique identifier of the message that describes the error (the verbiage of this message is displayed in the **Message Text** column).

FK Validation Summary

Navigate to the **Admin** menu and open **FK Validation Summary** to view foreign key validation errors associated with the objects defined in *Master Data*.

Description of Page

You can use **Table Name** to restrict errors to a specific object. If this field is left blank, all errors on all objects will be displayed.

The grid contains a separate row for each table and foreign key error combination.

The following information is displayed:

- **Table Name** is the name of the main table associated with the object.
- **Count** contains the number of records on this table that have this error.
- **Foreign Key Field Names 1 to 6** contain the names of the foreign keys contained on this table that have been found to be in error.
- **Foreign Key Values 1 to 6** contain the values within the foreign key fields that are found to be in error.

Note:

Most foreign keys on a table are obvious based on the field name. For example, if the output shows Table Name **CI_ADJ**, Count **100**, Foreign Key Field Name 1 **ADJ_TYPE_CD**, and Foreign Key Value 1 **SALES_INTEREST**, you know that there are 100 records in the Adjustment table referencing an invalid Adjustment Type Code of **SALES_INTEREST**. However, some foreign keys are multi-part keys and in viewing the Foreign Key names, the table they refer to may not be obvious. If it is not clear what table a foreign key is referring to, navigate to the Table page for the table reported in the error and view the Constraints tab. All the foreign keys being checked are listed there with their respective table names.

FK Validation Detail

This page is used to view foreign key validation errors of a given type associated with one of the objects defined in *Master Data*. This transaction is not intended to be invoked from the **Admin** menu. Rather, drill into the validation details from *FK Validation Summary*.

Description of Page

Use **Table Name** to specify the table you wish to view. The names and values of the foreign key fields on the table are displayed. The grid that follows contains the primary key values of this table's records that are in error. The following information is displayed:

- **Table Name** is the name of the main table.
- **FK Fields 1 to 6** are the names of the foreign keys contained in this table. Displayed alongside the key names are the values within these fields. These identify records on other tables to which this tables record is related. For example, the CI_ACCT_CHAR record identified by its displayed primary keys should be related to an Account record with the Account ID shown - it appears in this list only if there is something amiss with this relationship.

The Staging Tables

This section describes the objects into which your legacy data is mapped. For each object, we provide the following:

- A data model.
- An indication of which tables have system-assigned keys.
- The physical table names.
- The name of the batch control to submit to validate the object.
- The name of the program (and related batch control) that validates each table for referential integrity.
- The name of the program (and related batch control) that performs key assignment for each table.
- The name of the program (and related batch control) that inserts the table's rows into production from staging.
- Suggestions to assist in the conversion process.

Caution:

Recommendation. We recommend you read this document on a browser (or using Word under windows) so you can take advantage of the *Color Coding*.

Note:

Column details do not appear in this document. When you're ready to examine an object's tables, use the hyperlinks in the respective Table Names section to transfer to the *data dictionary*. The data dictionary will show you the required columns, the foreign keys (and their related tables), the source code of the program that validates the contents of the table, and a host of other information that will assist the conversion process.

Caution:

Look Up and Control Tables. In the data models that appear below, you will find a variety of entities that are classified as either a control table or a lookup table. Please refer to *Color Coding* for more information about how to recognize such an entity.

A Note About Programs in the Table Names Matrices

For each object described in the master data and transaction data sections, there is a "table names" section that includes a matrix listing the name of each table that is part of the maintenance object. Included in the matrix is information about the programs provided to perform object validation, referential integrity validation, key assignment and insertion. The following are some points about these programs:

- For maintenance objects that require *object validation*, an object validation program exists for the entire set of tables for the maintenance object. The **Object Validation Batch Control** column indicates the batch control used to submit the object validation. Refer to *Submitting Object Validation Programs* for more information. Drilling down on the hypertext allows you to see more information about the batch control, including the program associated with it.
- A separate *referential integrity batch validation* program exists for each table that requires one. As described in *Submitting Referential Integrity Validation Programs*, these programs are submitted using a driver supplied by the system where the batch code for the appropriate table is provided. (The driver then executes the program whose name matches the batch code). The **Referential Integrity Validation Batch Control** column indicates the table's batch control / program name.
- One *key assignment* program exists for the parent table for the maintenance object. As described in *Submitting Key Assignment Programs*, these programs are submitted using a driver supplied by the system where the batch code for the appropriate table is provided. (The driver then executes the program whose name matches the batch code). The **Key Assignment Batch Control** column indicates the table's batch control / program name.
- An *insertion program* exists for every table for the maintenance object. As described in *Submitting Insertion Programs*, these programs are submitted using a driver supplied by the system where the batch code for the appropriate table is provided. (The driver then executes the program whose name matches the batch code). The **Insertion Batch Control** column indicates the table's batch control / program name.

Master Data

This section describes the various "master data" objects (e.g., person, account etc.) that must be created before you can convert transaction data.

Note:

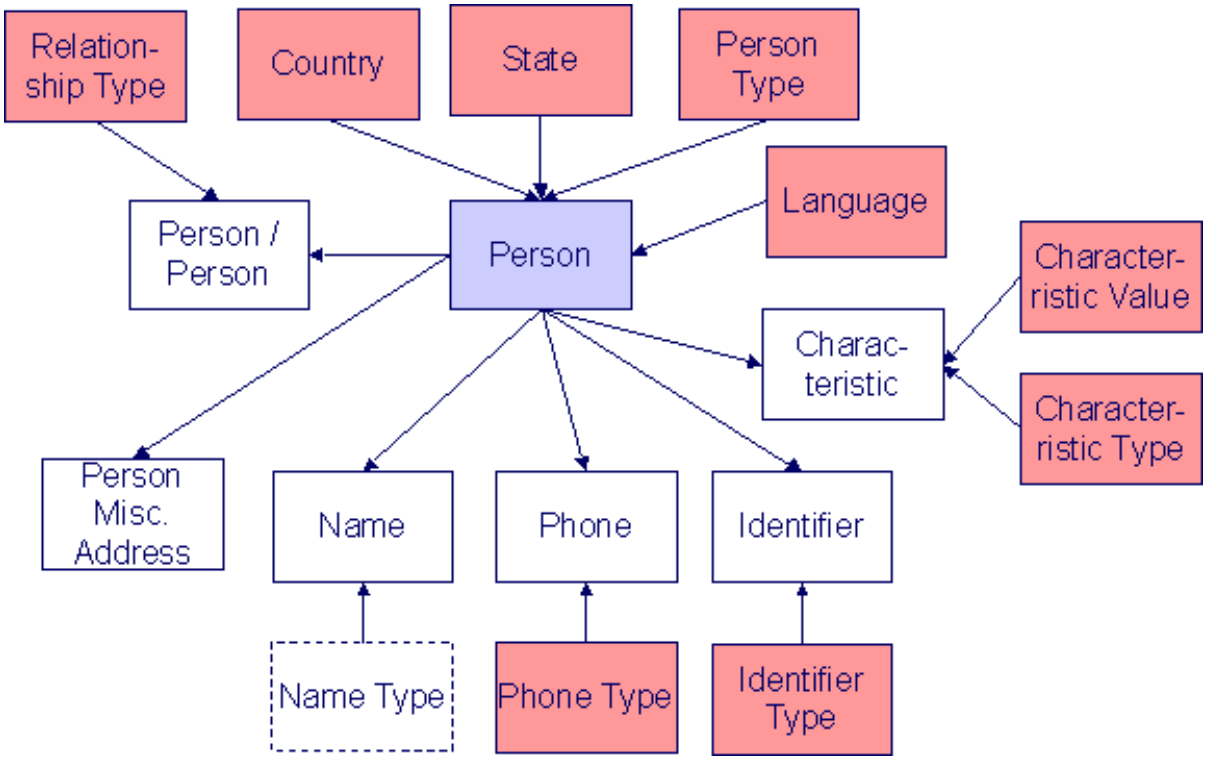
Key Assignment Dictates The Order Of Conversion. The following contents are listed in the order in which the objects should be converted in order to maintain referential integrity.

Person

This section describes the person object. Refer to *Account* for details about the account object.

Person Data Model

The following data model illustrates the person object.



Person Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Person	*CI_PER*	Yes *CI_PER_K*	*VAL-PER*		CIPVPERK	CIPVPERI
Name	*CI_PER_NAME*	No. The key is PER_ID plus a sequence number.		CIPVPNMV		CIPVPNMI
Person / Person	*CI_PER_PER*	No. The key is PER_ID1, PER_ID2, relationship type and start date.		CIPVPPEV		CIPVPPEI
Phone	*CI_PER_PHONE*	No. The key is PER_ID plus phone type.		CIPVPPHV		CIPVPPHI

Identifier	*CI_PER_ID*	No. The key is PER_ID plus identifier type.		CIPVPIDV		CIPVPIDI
Characteristic	*CI_PER_CHAR*	No. The key is PER_ID plus an edate and a char type.		CIPVPRCV		CIPVPRCI
Miscellaneous Address	*CI_PER_ADDR_SEQ*	No. The key is PER_ID plus a sequence number.		CIPVPSAV		CIPVPSAI

Person Suggestions

A person must have at least one row on the name table and at least one of the names must be marked as being the primary name.

A person must have at least one row on the identity table and at least one of the identities must be marked as being the primary ID.

The country and state are only necessary if the person has an override mailing address.

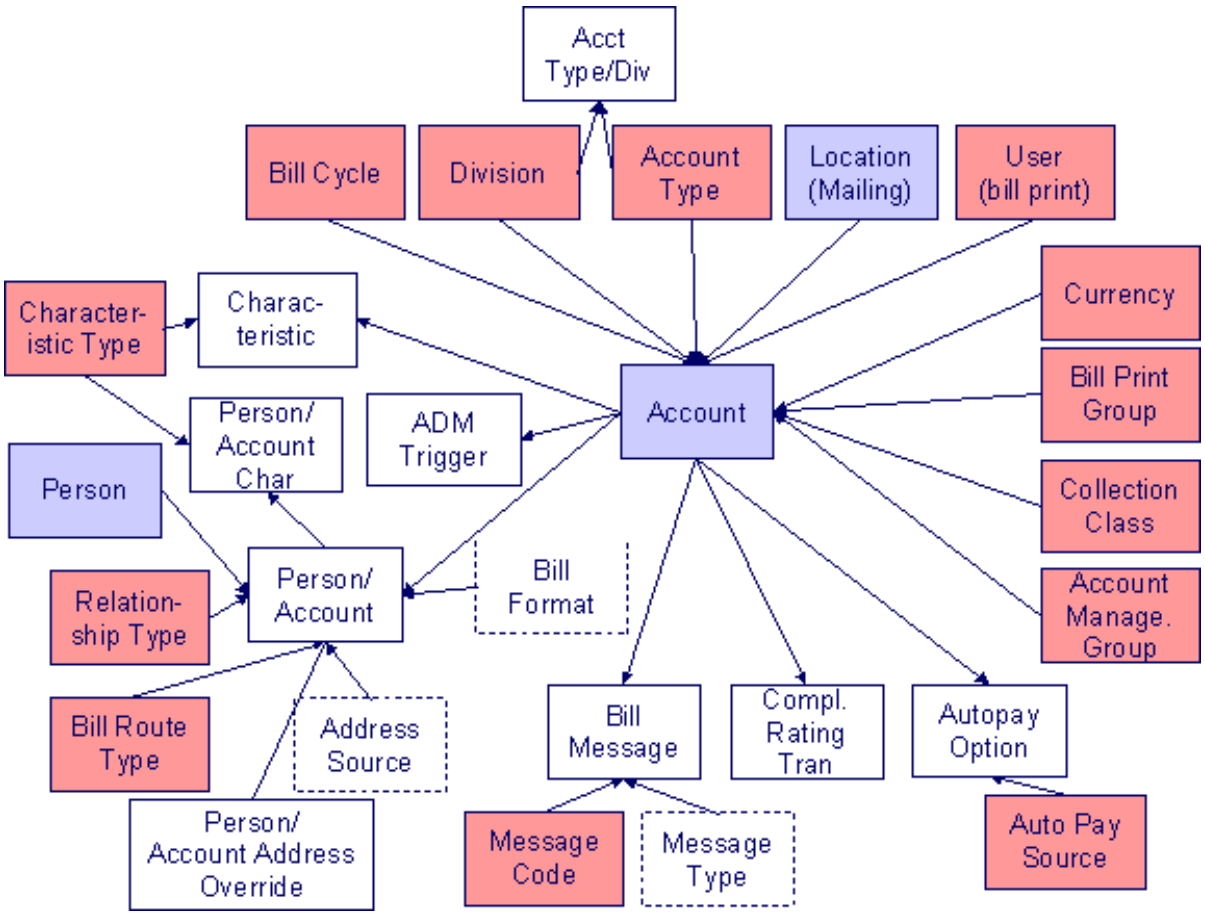
This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Account

Each customer must have a person and an account object. This section describes the account object, refer to *Person* for details about the person object.

Account Data Model

The following data model illustrates the account object.



Account Table Names

Data Model Name	***Table Name***	***Generated Keys***	***Object Validation Batch Control***	***Referential Integrity Validation Batch Control***	***Key Assignment Batch Control***	***Insertion Batch Control***
Account	*CI_ACCT*	Yes *CI_ACCT_K*	*VAL-ACCT*		CIPVACCK	CIPVACCI See Optional FK Note below.
Bill Message	*CI_ACCT_MSG*	No. The key is account ID plus bill message code.		CIPVMSGV		CIPVMSGI
Autopay Option	*CI_ACCT_APAY*	Yes *CI_ACCT_APAY_K*		CIPVAAPV	CIPVAAPK *Has dependencies*	CIPVAAPI
Characteristic	*CI_ACCT_CHAR*	No. The key is ACCT_ID plus		CIPVACHV		CIPVACHI

		an edate and a char type.				
Person/Account	*CI_ACCT_PER*	No. The key is account ID plus person ID.		CIPVACPV		CIPVACPI
Person / Account Char	*CI_ACCT_PER_CHAR*	No. The key is account ID plus person ID plus an edate and a char type.		CIPVAPCV		CIPVAPCI
Person/Account Address Override	*CI_PER_ADDR_OVRD*	No. The key is Account ID plus Person ID		CIPVPAOV		CIPVPAOI
Compliance Rating Transaction	*CI_CR_RAT_HIST*	Yes *CI_CR_RAT_HIST_K*		CIPVCRTV	CIPVCRRK *Has dependencies*	CIPVCRTI
ADM Trigger	*CI_ADM_RVW_SCH*	No. The key is account ID plus date		CIPVARSV		CIPVARSI

Account Suggestions

An account must have at least one row on the account / person table and at least one account / person must be marked as being the main customer. Please see column notes for the account / person table for inter-field validation in respect of the various switches (e.g., if main customer switch is on, then the person must also be financially responsible).

We recommend storing an ADM trigger (*CI_ADM_RVW_SCH*) for every account where the trigger date is the conversion date. This will cause the account to be reviewed by the *overdue monitor* when it next runs. We have supplied a dedicated batch process for this purpose that simply inserts a row in this table with the review date set equal to the current date. This will ensure that all converted accounts are reviewed after they are inserted into production. This program is named CIPVADMB and goes by the batch control ID of **CNV-ADM**.

If your legacy system has the equivalent of a compliance rating, you should create compliance rating transactions. The values you create need to be consistent with the base and threshold compliance rating on the installation record. Refer to the account user documentation for more information.

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

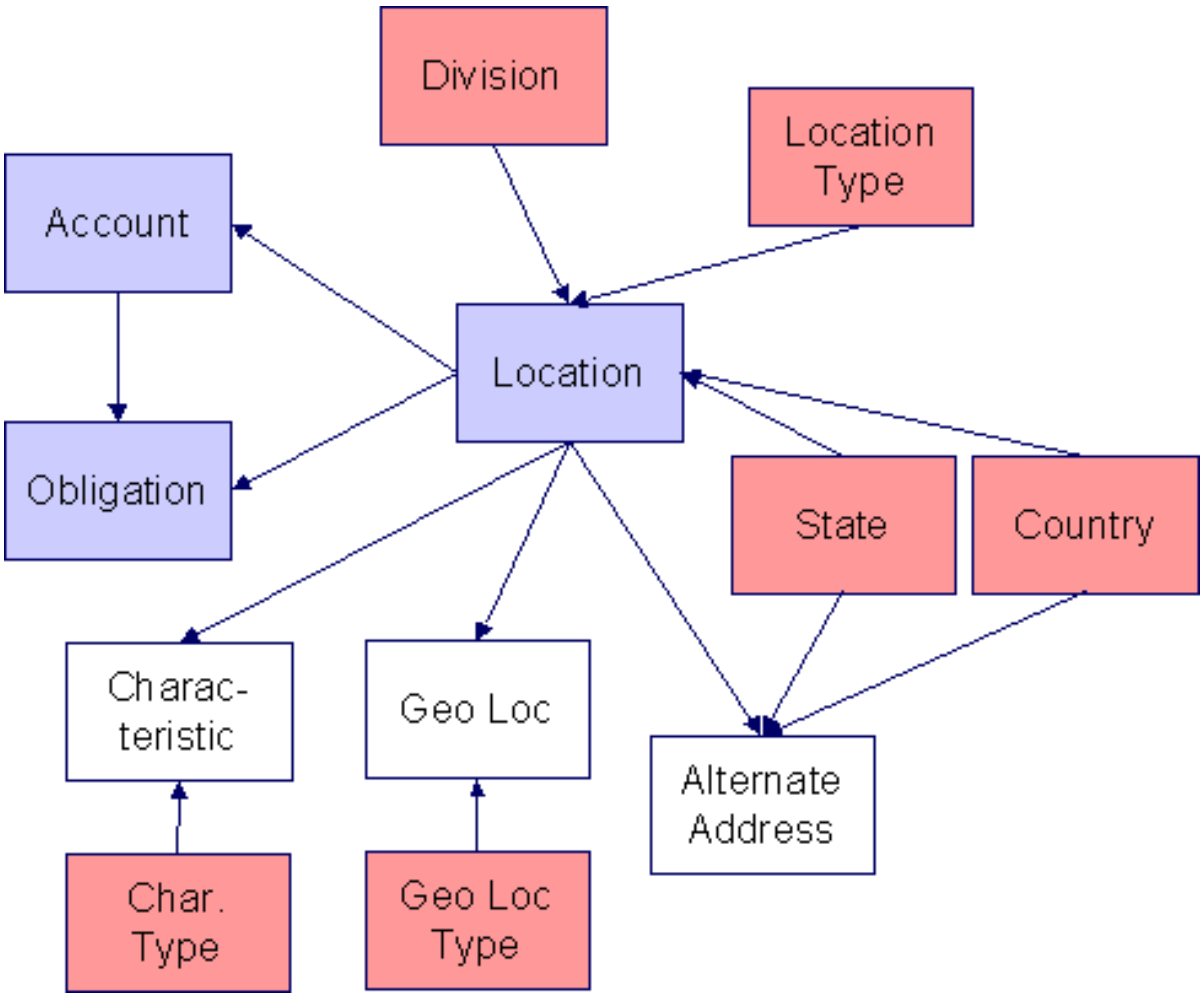
Optional FK Note

Account has an optional foreign key MAILING_PREM_ID to the Location (CI_PREM) table. If your implementation is planning to convert accounts, but not locations the key generation program for Location must still be run. Refer to *The Big Picture of Key Assignment* for more information.

Location

Location Data Model

The following data model illustrates the location object.



Location Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Location	*CI_PREM*	Yes *CI_PREM_K*	*VAL-PREM*		CIPVPRMK *Has dependencies*	CIPVPRMI
Characteristic	*CI_PREM_CHAR*	No. The key is PREM_ID plus		CIPVPCHV		CIPVPCHI

		an edate and a char type.				
Geo Loc	*CI_PREM_GEO*	No. The key is PREM_ID plus geo loc type.		CIPVPGOV		CIPVPGOI
Alternate Address	*CI_PRM_ALT_ADDR*	Yes *CI_PRM_ALT_ADDR_K*		CIPVAPAV	CIPVAPAK *Has dependencies*	CIPVAPAI

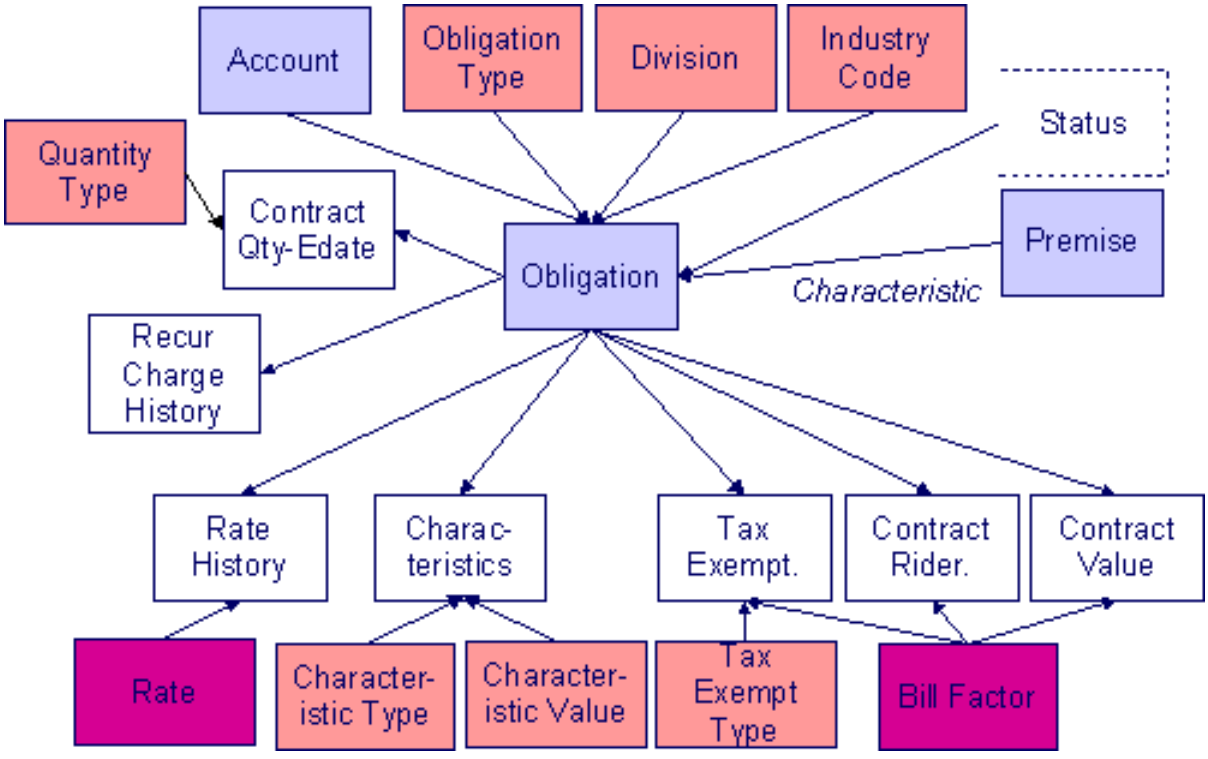
Location Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Obligation

Obligation Data Model

The following data model illustrates the obligation object.



Obligation Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Obligation	*CI_SA*	Yes *CI_SA_K*	VAL-OBLG		CIPVSVAK *Has dependencies*	CIPVSVAI See Optional FK Note below.
Characteristic	*CI_SA_CHAR*	No. The key is obligation_ID plus an edate and a char type.		CIPVSACV		CIPVSACI
Contract Quantity Edate	*CI_SA_CONT_QTY*	No. The key is obligation ID plus quantity type plus edate.		CIPVSAQV		CIPVSAQI
Message	*CI_SA_MSG*	No. The key is obligation ID plus Bill message code.		CIPVSMGV		CIPVSMGI
Recurring Charge	*CI_SA_RCHG_HIST*	No. The key is obligation ID plus edate.		CIPVSARV		CIPVSARI
Rate History	*CI_SA_RS_HIST*	No. The key is obligation ID plus edate.		CIPVSAHV		CIPVSAHI
Tax Exempt	*CI_SA_CONTERM* - this table is also used for the next 2 entities, the key contains CONTERM_TYPE_FLG that controls the entity	No. This key is obligation ID plus CONTERM_TYPE_FLG plus BF_CD plus START_DT		CIPVSAOV		CIPVSAOI
Contract Rider	*CI_SA_CONTERM* - this table is also used for the previous and next entities, the key contains CONTERM_TYPE_FLG that controls the entity	No. This key is obligation ID plus CONTERM_TYPE_FLG plus BF_CD plus START_DT		CIPVSAOV		CIPVSAOI

Contract Value	*CI_SA_CONTERM* - this table is also used for the previous 2 entities, the key contains CONTERM_TYPE_FLG that controls the entity	No. This key is obligation ID plus CONTERM_TYPE_FLG plus BF_CD plus START_DT		CIPVSAOV		CIPVSAOI

Obligation Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

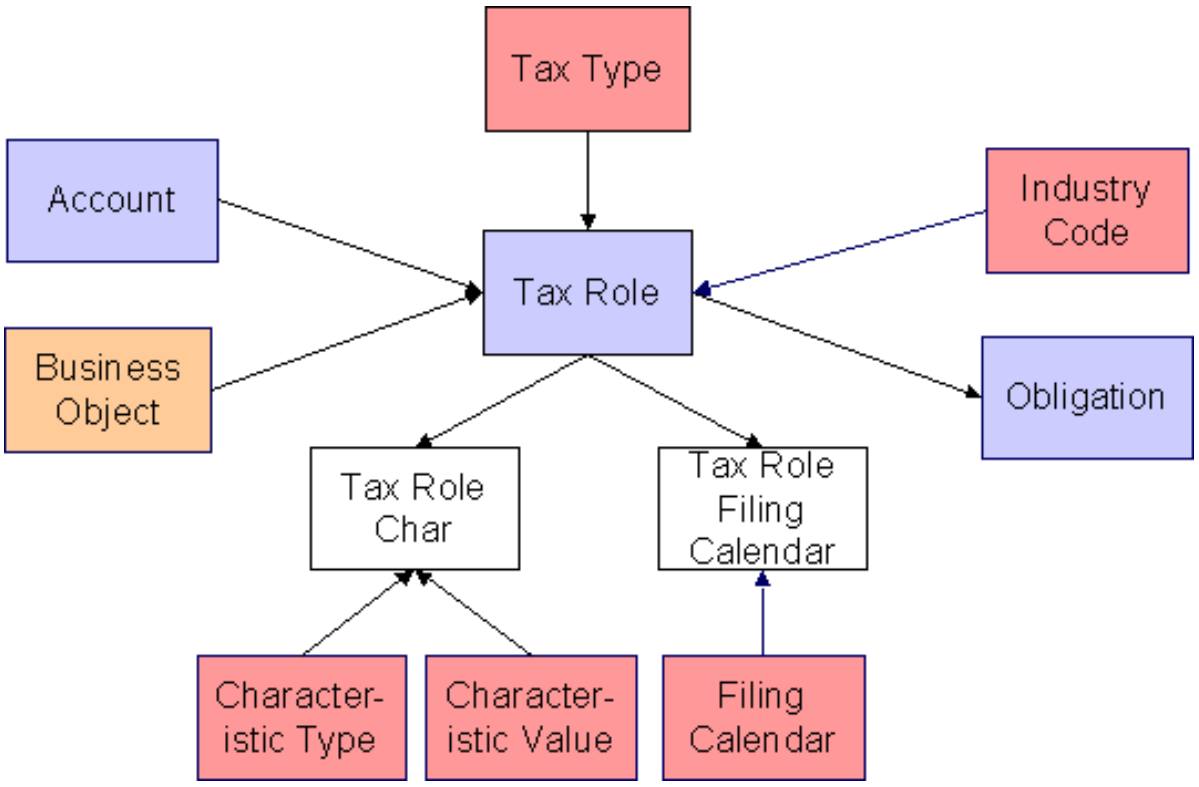
Optional FK Note

Account has an optional foreign key CHAR_PREM_ID to the Location (CI_PREM) table and an optional foreign key TAX_ROLE_ID to the Tax Role (CI_TAX_ROLE) table. If your implementation is planning to convert obligations, but not locations or tax roles, the key generation programs for Location and Tax Role must still be run. Refer to *The Big Picture of Key Assignment* for more information.

Tax Role

Tax Role Data Model

The following data model illustrates the tax role object.



Tax Role Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Tax Role	*CI_TAX_ROLE*	Yes. *CI_TAX_ROLE_K*	*VAL-TAXR*	CIPVTXRV	CIPVTXRK	CIPVTXRI
Tax Role Filing Calendar	*CI_TAX_ROLE_CAL*	No. The key is Tax Role ID and Effective Date.				
Tax Role Characteristic	*CI_TAX_ROLE_CHAR*	No. The key is Tax Role ID, Characteristic Type, and Effective Date.		CIPVTXCV		CIPVTXCI

Tax Role Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Transaction Data

This section describes the tables in which your transaction data (e.g., bills, payments, customer contacts, etc.) resides.

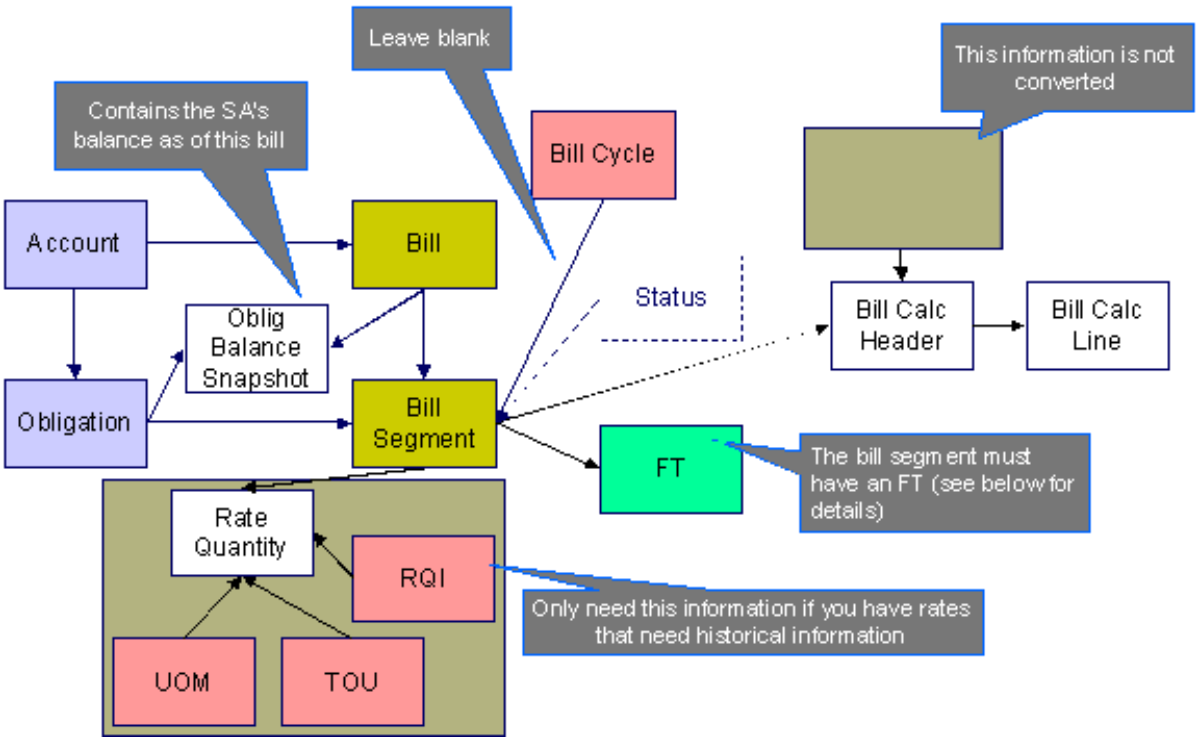
Contents

Bill

Bill Data Model

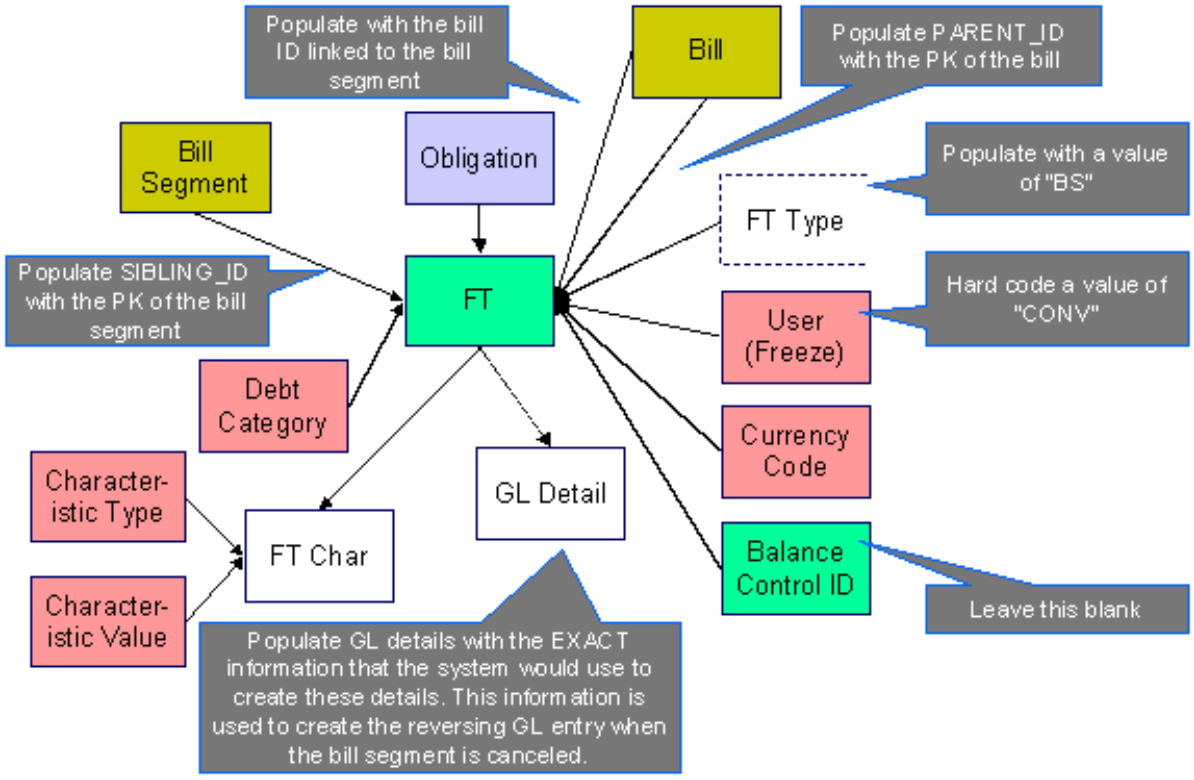
Bill - Main

The following data model illustrates the bill object.



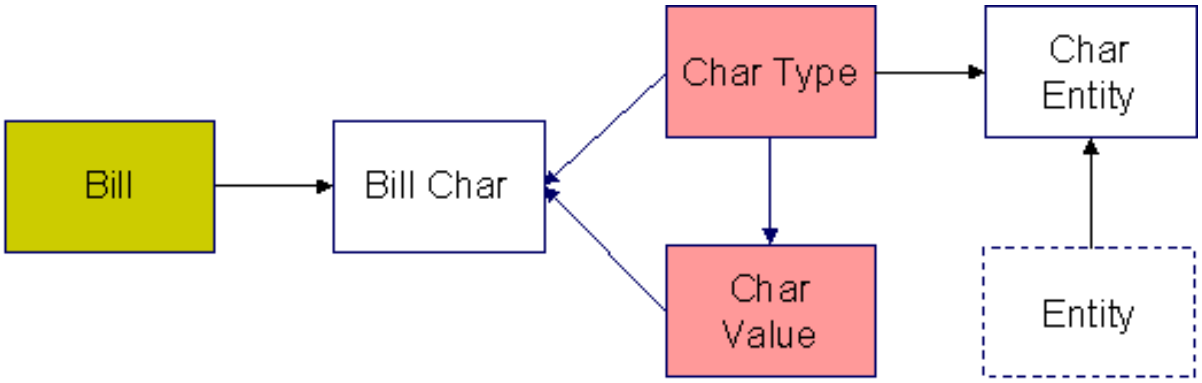
Bill - FT

The following data model illustrates the FT that must be associated with a bill segment.



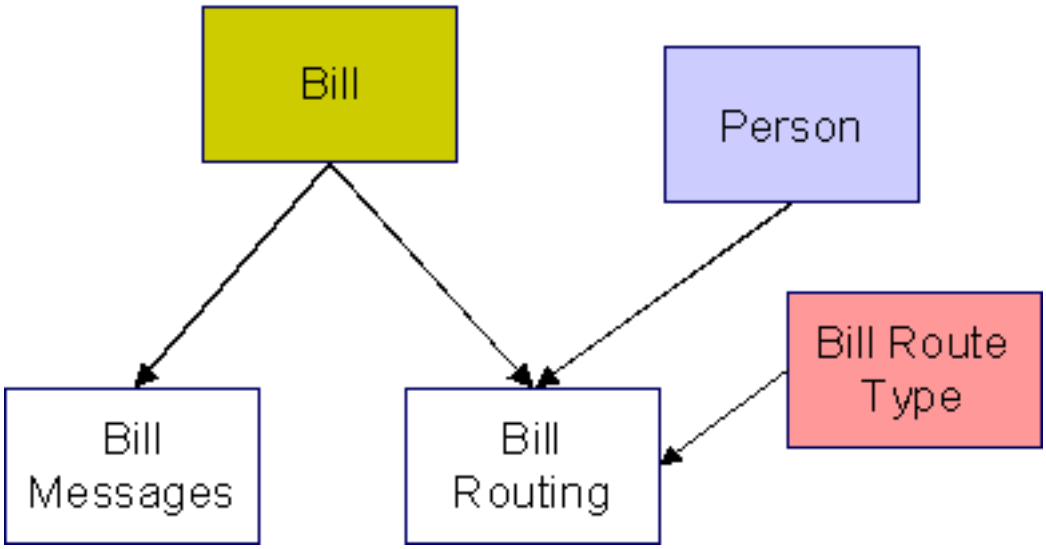
Bill Characteristics

The following data model illustrates Bill Characteristics.



Bill Messages and Routing

The following data model illustrates Bill Messages and Bill Routing.



Bill Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Bill	*CI_BILL*	Yes *CI_BILL_K*	CIPVBLLV	CIPVBILK *Has dependencies*	CIPVBLLI
Obligation Balance Snapshot	*CI_BILL_SA*	No. The key is bill ID plus obligation ID.	CIPVBSAV		CIPVBSAI
Bill Segment	*CI_BSEG*	Yes *CI_BSEG_K*	CIPVSEGV	CIPVBSGK *Has dependencies*	CIPVSEGI
Calc Header	*CI_BSEG_CALC*	No. The key is bill segment id and a sequence number	CIPVBSCV		CIPVBSCI
Calc Lines	*CI_BSEG_CALC_LN*	No. The key is bill segment id, the header sequence number, and a sequence number	CIPVBSLV		CIPVBSLI
Read Detail	*CI_BSEG_READ*	No. The key is bill segment id and a sequence number.	CIPVSRRV		CIPVSRRI
Asset Detail	*CI_BSEG_ITEM*	No. The key is bill segment id and a sequence number	CIPVBSIV		CIPVBSII

Rate Quantity	*CI_BSEG_SQ*	No. The key is bill segment id, uom code, tou code and RQI code	CIPVSQTV		CIPVSQTI
FT (financial transaction)	*CI_FT*	Yes *CI_FT_K*	CIPVFTFV	CIPVFTXK *Has dependencies*	CIPVFTFI
FT Characteristic	*CI_FT_CHAR*	No. The key is FT id, char type code and a sequence number	CIPVFTCV		CIPVFTCI
FT GL (FT general ledger)	*CI_FT_GL*	No. The key is FT id and a GL sequence number	CIPVFTGV		CIPVFTGI
Characteristics	*CI_BILL_CHAR*	No. The key is bill id, char type code and a sequence number	CIPVBCHV		CIPVBCHI
Bill Messages	*CI_BILL_MSGS*	No. The key is bill id and bill message code.	CIPVBLMV		CIPVBLMI
Bill Routing	*CI_BILL_ROUTING*	No. The key is bill id and a sequence number	CIPVBLRV		CIPVBLRI

Bill Suggestions

Most companies have found it impossible to load bill segment item, bill calc header and lines with sufficient information and therefore these tables are not populated. See the comments in the above ERD's for more information.

Please populate the columns on the FT that's associated with the bill segment as follows:

- CUR_AMT should be set equal to the bill segment amount
- PAY_AMT should be set equal to the bill segment amount
- CRE_DTTM should be set equal to the bill segment end date / time
- FREEZE_SW should be "Y"
- FREEZE_DTTM should be set equal to the bill segment end date / time
- ARS_DT should be set equal to the bill segment end date
- CORRECTION_SW should be "N"
- REDUNDANT_SW should be "N"
- NEW_DEBIT_SW should be "N"
- NOT_IN_ARS_SW should be set to "N"
- SHOW_ON_BILL_SW should be set to "N"
- ACCOUNTING_DT should be set to the current date
- SCHED_DISTRIB_DT should be left blank
- CURRENCY_CD should be the currency on the installation record
- BAL_CTL_GRP_ID should be left blank
- XFERRED_OUT_SW should be set to "Y"
- PARENT_ID should be set to the bill ID

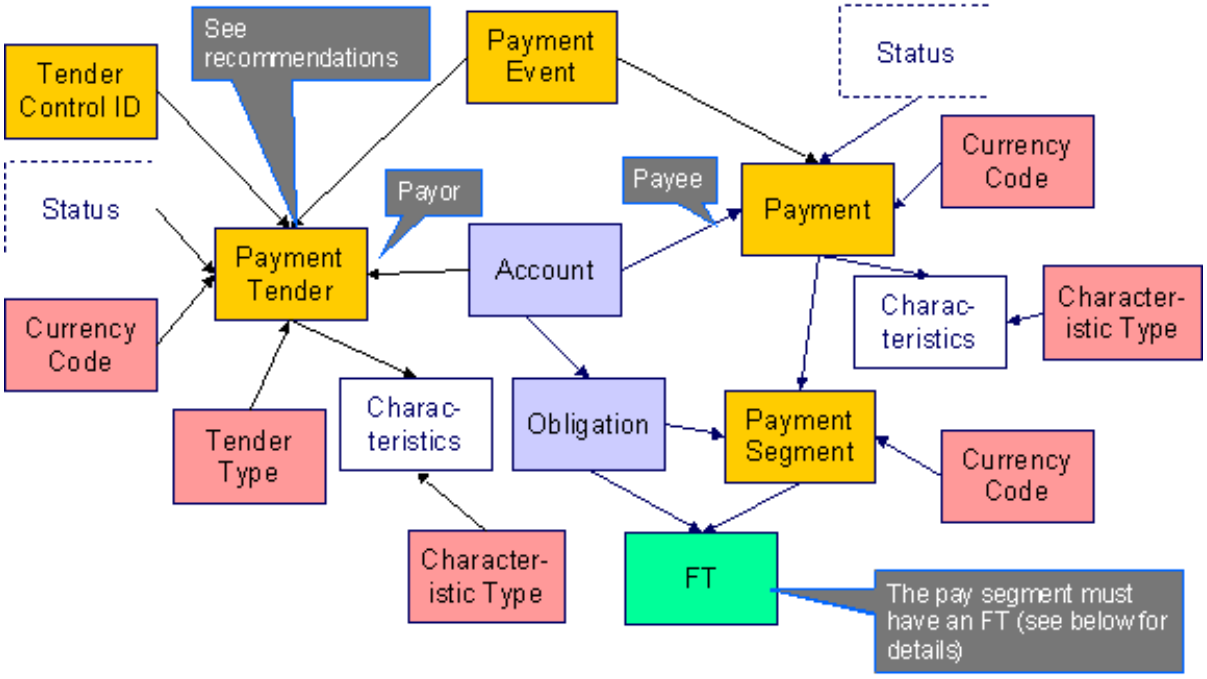
- SIBLING_ID should be set to the bill segment ID
- Do NOT create any GL details for the FT. If GL details are converted, ensure they are populated with the EXACT information the system would use to create them. This information is used to create the reversing GL entry when the bill segment is canceled.

Payment

Payment Data Model

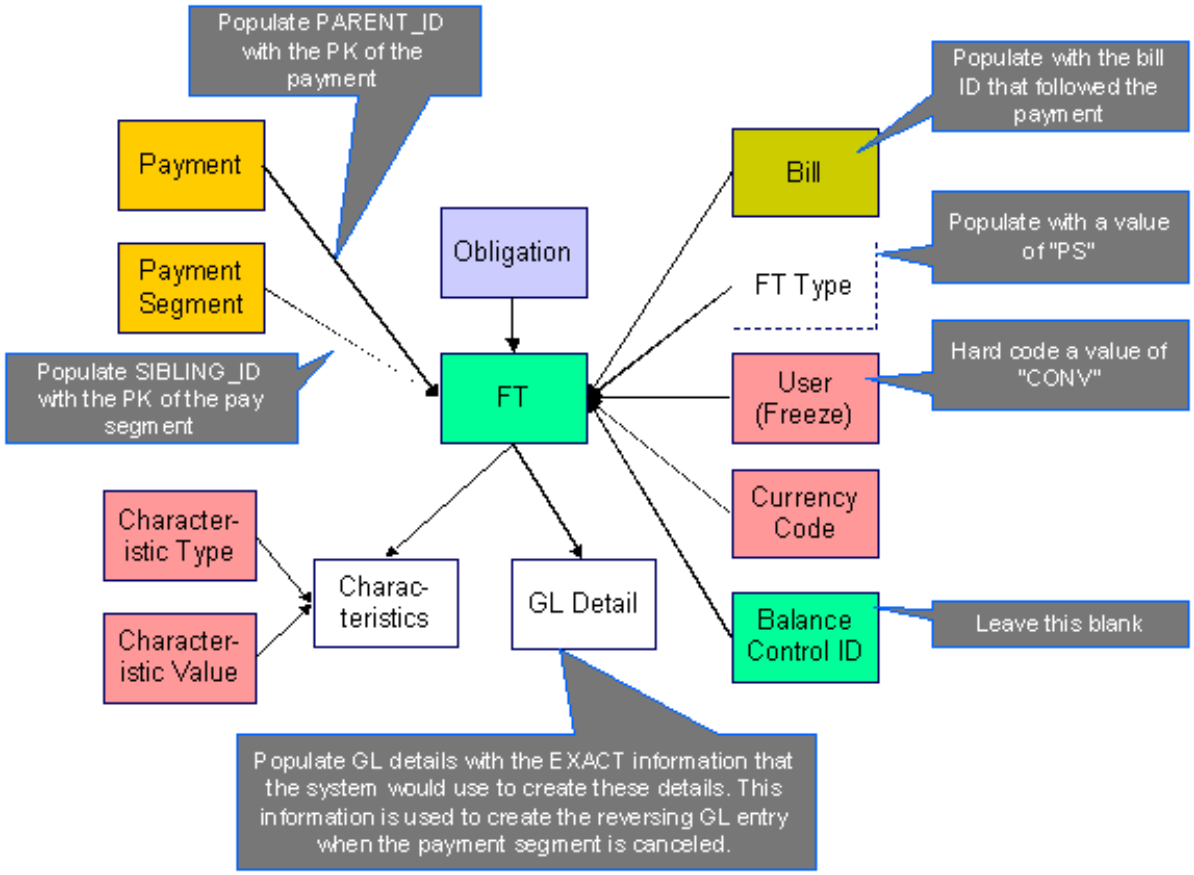
Payment - Main

The following data model illustrates the payment object.



Payment - FT

The following data model illustrates the FT that must be associated with a payment segment.



Payment Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Payment	*CI_PAY*	Yes*CI_PAY_K*	CIPVPAYV	CIPVPAYK *Has dependencies*	CIPVPAYI
Payment Characteristic	*CI_PAY_CHAR*	No. The key is PAY_ID, plus a sequence number and a char type	CIPVPYCV		CIPVPYCI
Payment Event	*CI_PAY_EVENT*	Yes*CI_PAY_EVENT_K*		CIPVPYEK *Has dependencies*	CIPVPYEI
Payment Tender	*CI_PAY_TNDR*	Yes*CI_PAY_TNDR_K*	CIPVTNDV	CIPVTNDK *Has dependencies*	CIPVTNDI
Payment Tender Characteristic	*CI_PAY_TNDR_CHAR*	No. The key is PAY_TENDER_ID, plus a sequence	CIPVTNCV		CIPVTNCI

		number and a char type			
Payment Segment	*CI_PAY_SEG*	Yes *CI_PAY_SEG_K*	CIPVPSGV	CIPVPSGK *Has dependencies*	CIPVPSGI
FT (financial transaction)	*CI_FT*	Yes *CI_FT_K*	CIPVFTFV	CIPVFTXK *Has dependencies*	CIPVFTFI See Optional FK Note below.
FT Characteristic	*CI_FT_CHAR*	No. The key is FT id, char type code and a sequence number	CIPVFTCV		CIPVFTCI
FT GL (FT general ledger)	*CI_FT_GL*	No. The key is FT id and a GL sequence number	CIPVFTGV		CIPVFTGI

Payment Suggestions

The SEQ_NUM field on the payment should be left blank. This is part of the primary key for the Payment Event Distribution Details, which are not part of the conversion process.

We recommend that you use the system to create a single deposit control and link to it a single tender control using the PRODUCTION tables. The tender control should reference a tender source of "conversion". Use the prime key of the tender control as the foreign key on the tenders that you insert into the STAGING tables. This means you will have an invalid foreign key relationship on CI_PAY_TNDR (it will reference a tender control that doesn't exist).

After converting the payments:

- Re-access the tender control in production and enter the appropriate amounts (per tender type) to balance the tender control.
- Re-access the deposit control in production and enter the appropriate amounts to balance the deposit control.

Please populate the columns on the FT that's associated with the payment segment as follows:

- CUR_AMT should be set equal to the payment segment amount
- PAY_AMT should be set equal to the payment segment amount
- CRE_DTTM should be set equal to the payment segment date / time
- FREEZE_SW should be "Y"
- FREEZE_DTTM should be set equal to the payment segment date / time
- ARS_DT should be set equal to the payment segment date
- CORRECTION_SW should be "N"
- REDUNDANT_SW should be "N"
- NEW_DEBIT_SW should be "N"
- NOT_IN_ARS_SW should be set to "N"
- SHOW_ON_BILL_SW should be set to "N" on all payments other than payments that have been received since the last bill. For recent payments that you want to show on the next bill, this switch must be "Y"
- ACCOUNTING_DT should be set to the current date
- SCHED_DISTRIB_DT should be left blank
- CURRENCY_CD should be the currency on the installation record
- BAL_CTL_GRP_ID should be left blank
- XFERRED_OUT_SW should be set to "Y"
- PARENT_ID should be set to the payment ID

- SIBLING_ID should be set to the payment segment ID
- Do NOT create any GL details for the FT. If GL details are converted, ensure they are populated with the EXACT information the system would use to create them. This information is used to create the reversing GL entry when the payment segment is canceled.

Optional FK Note

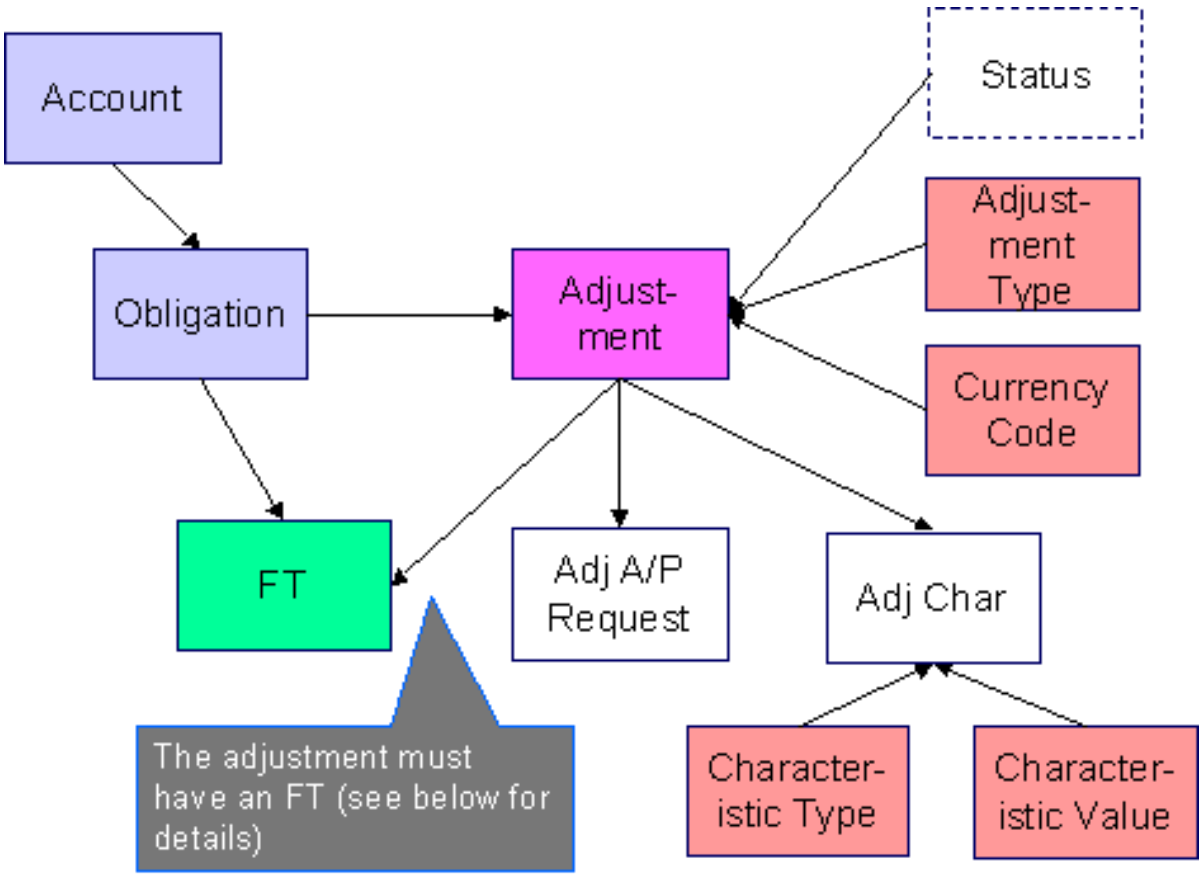
FTs for adjustments have an optional foreign key BILL_ID to the Bill (CI_BILL) table. If your implementation is planning to convert financial transactions for adjustments, but does not plan to convert bills, the key generation program for Bill must still be run. Refer to *The Big Picture of Key Assignment* for more information.

Adjustment

Adjustment Data Model

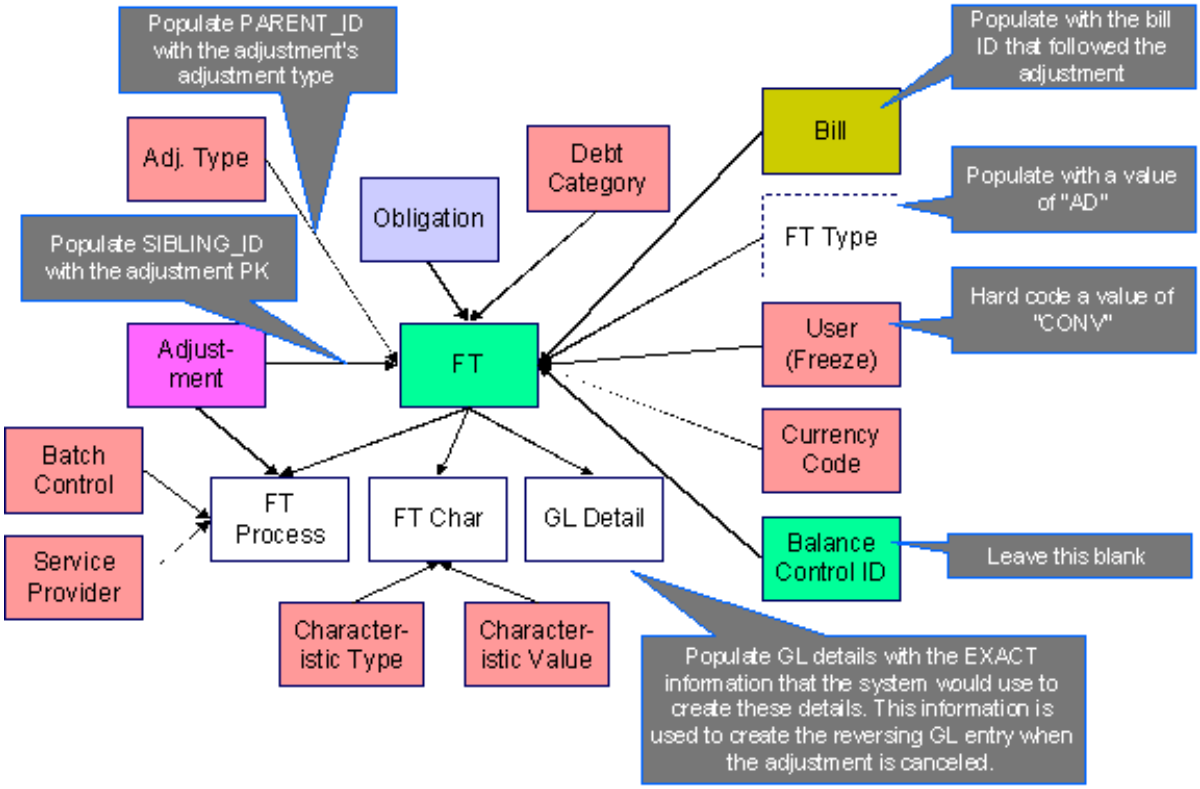
Adjustment - Main

The following data model illustrates the adjustment object.



Adjustment - FT

The following data model illustrates the FT that must be associated with an adjustment segment.



Adjustment Table Names

Data Model Name	***Table Name***	***Generated Keys***	***Referential Integrity Validation Batch Control***	***Key Assignment Batch Control***	***Insertion Batch Control***
Adjustment	*CI_ADJ*	Yes *CI_ADJ_K*	CIPVADJV	CIPVADJK *Has dependencies*	CIPVADJI
Adjustment A/P Request	*CI_ADJ_APREQ*	Yes *CI_ADJ_APREQ_K*	CIPVAPRV	CIPVAPRK *Has dependencies*	CIPVAPRI
FT (financial transaction)	*CI_FT*	Yes *CI_FT_K*	CIPVFTFV	CIPVFTXK *Has dependencies*	CIPVFTFI See Optional FK Note below.
FT Characteristic	*CI_FT_CHAR*	No. The key is FT id, char type code and a sequence number	CIPVFTCV		CIPVFTCI
FT GL (FT general ledger)	*CI_FT_GL*	No. The key is FT id and a GL sequence number	CIPVFTGV		CIPVFTGI

FT Process	*CI_FT_PROC*	No. The key is FT id and a sequence number	CIPVFTPV		CIPVFTPI
Adjustment Characteristic	*CI_ADJ_CHAR*	No. The key is adjustment id, char type code and a sequence number	CIPVADCV		CIPVADCI

Adjustment Suggestions

Please populate the columns on the FT that's associated with the adjustment as follows:

- CUR_AMT should be set equal to the adjustment amount
- PAY_AMT should be set equal to the adjustment amount
- CRE_DTTM should be set equal to the adjustment date / time
- FREEZE_SW should be "Y"
- FREEZE_DTTM should be set equal to the adjustment date / time
- ARS_DT should be set equal to the adjustment date
- CORRECTION_SW should be "N"
- REDUNDANT_SW should be "N"
- NEW_DEBIT_SW should be "N"
- NOT_IN_ARS_SW should be set to "N"
- SHOW_ON_BILL_SW should be set to "N" on all adjustments other than adjustments that have been generated since the last bill. For recent adjustments that you want to show on the next bill, this switch must be "Y"
- ACCOUNTING_DT should be set to the current date
- SCHED_DISTRIB_DT should be left blank
- CURRENCY_CD should be the currency on the installation record
- BAL_CTL_GRP_ID should be left blank
- XFERRED_OUT_SW should be set to "Y"
- PARENT_ID should be set to the adjustment's adjustment type
- SIBLING_ID should be set to the adjustment ID
- Do NOT create any GL details for the FT. If GL details are converted, ensure they are populated with the EXACT information the system would use to create them. This information is used to create the reversing GL entry when the adjustment is canceled.

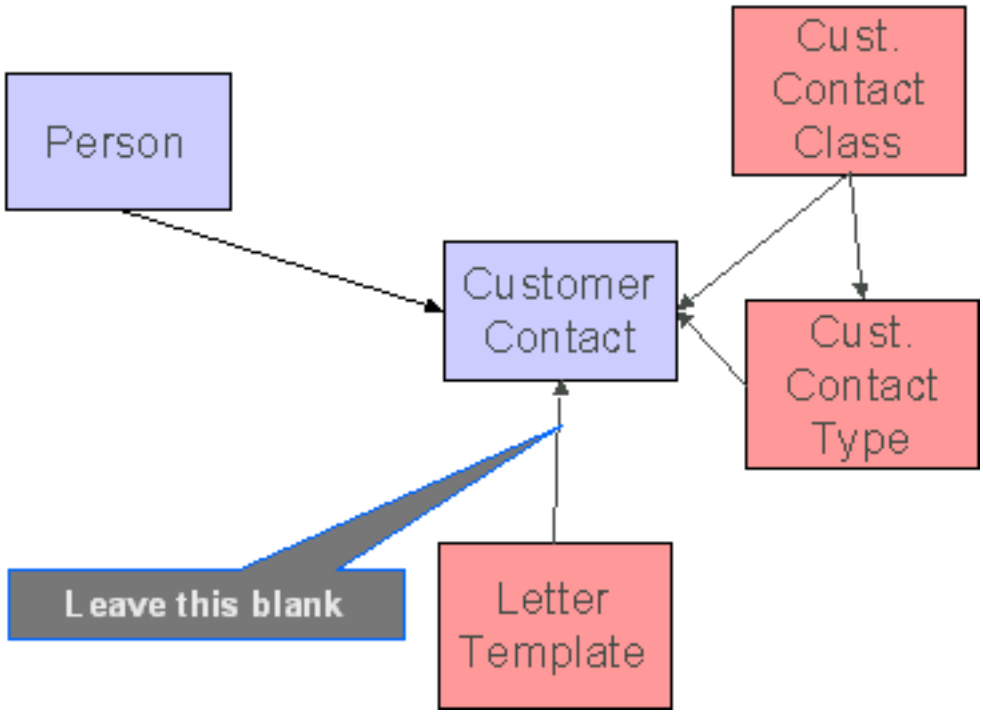
Optional FK Note

FTs for adjustments have an optional foreign key BILL_ID to the Bill (CI_BILL) table. If your implementation is planning to convert financial transactions for adjustments, but does not plan to convert bills, the key generation program for Bill must still be run. Refer to *The Big Picture of Key Assignment* for more information.

Customer Contact

Customer Contact Data Model

The following data model illustrates the Customer Contact object.



Customer Contact Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Customer Contact	*CI_CC*	Yes *CI_CC_K*	CIPVCSCV	CIPVCCTK *Has dependencies*	CIPVCSCI

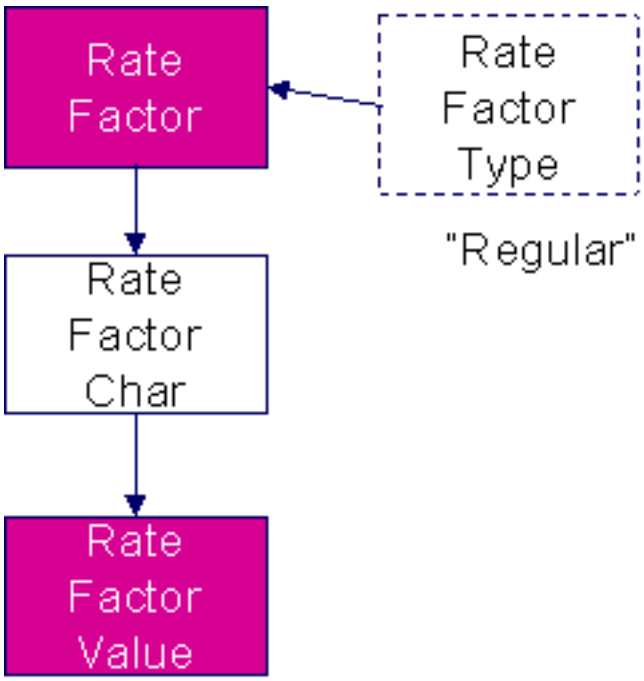
Customer Contact Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Rate Factor Value

Rate Factor Value Data Model

The following data model illustrates the rate factor objects.



Rate Factor Value Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Rate Factor Value	*CI_BF_VAL*	No. The key is BF_CD plus CHAR_TYPE_CD plus CHAR_VAL plus EFFDT	CIPVBFVV		CIPVBFVI (Not threadable)

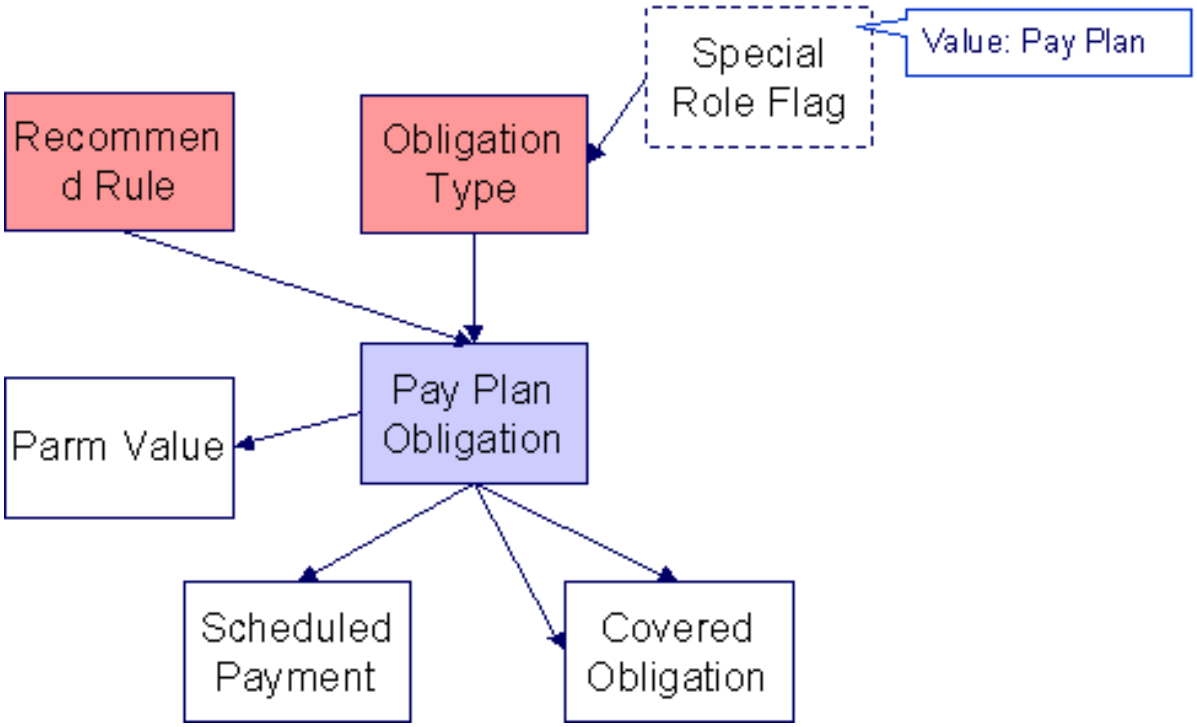
Rate Factor Value Suggestions

N/A.

Payment Plan

Payment Plan Data Model

The following data model illustrates the Payment Plan objects.



Payment Plan Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Covered Obligation	*CI_NB_SA*	No. The key is SA_ID plus CVRD_SA_ID	CIPVNBSV		CIPVNBSI
Scheduled Payments	*CI_NB_SCHED_PAY*	Yes *CI_NB_SCHED_PAY_K*	CIPVNSPV	CIPVNSPK *Has dependencies*	CIPVNSPI
Pay Plan Parameters	*CI_SA_NB_PARM*	No. The key is SA_ID plus Sequence Number.	CIPVNPMV		CIPVNPMI

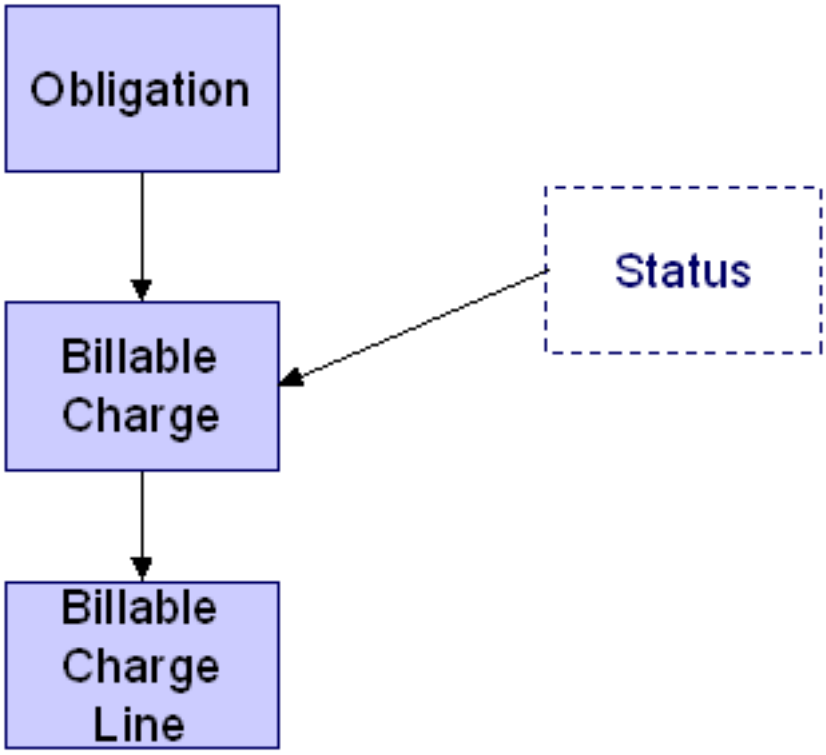
Payment Plan Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Billable Charge

Billable Charge Data Model

The following data model illustrates the Billable Charge objects.



Billable Charge Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Billable Charge	*CI_BILL_CHG*	Yes. *CI_BILL_CHG_K*	*VAL-BCHG*	CIPVBCGV	CIPVBCGK	CIPVBCGI
Billable Charge Line	*CI_B_CHG_LINE*	No. The key is billable charge id and a sequence number		CIPVBCLV		CIPVBCLI

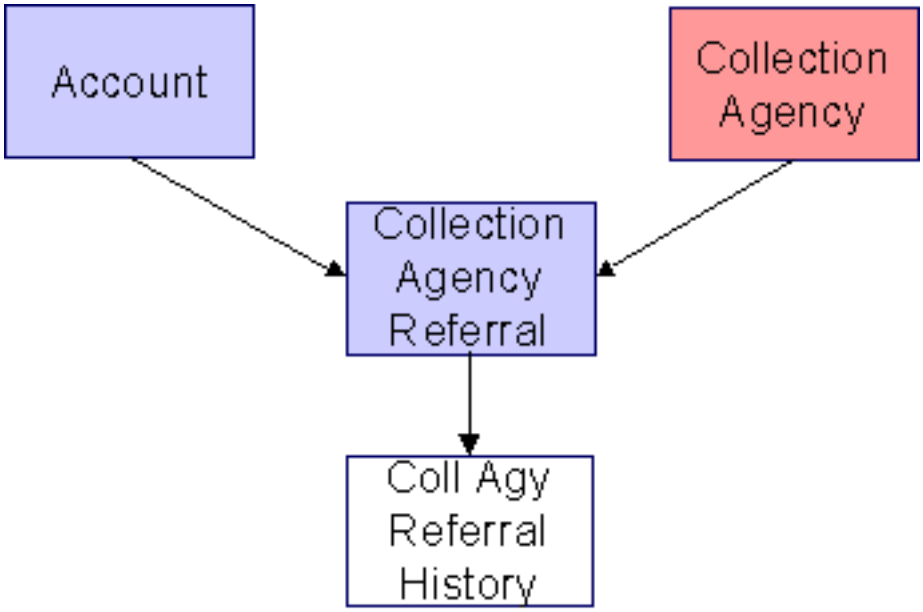
Billable Charge Suggestions

N/A

Collection Agency Referral

Collection Agency Referral Data Model

The following data model illustrates the Collection Agency Referral object.



Collection Agency Referral Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Collection Agency Referral	*CI_COLL_AGY_REF*	Yes. *CI_COLL_AGY_REF_K*	CIPVCARV	CIPVCARK	CIPVCARI
Collection Agency Referral History	*CI_COLL_AGY_HIS*	No. The key is collection agency referral id and characteristic type code	CIPVARHV		CIPVARHI

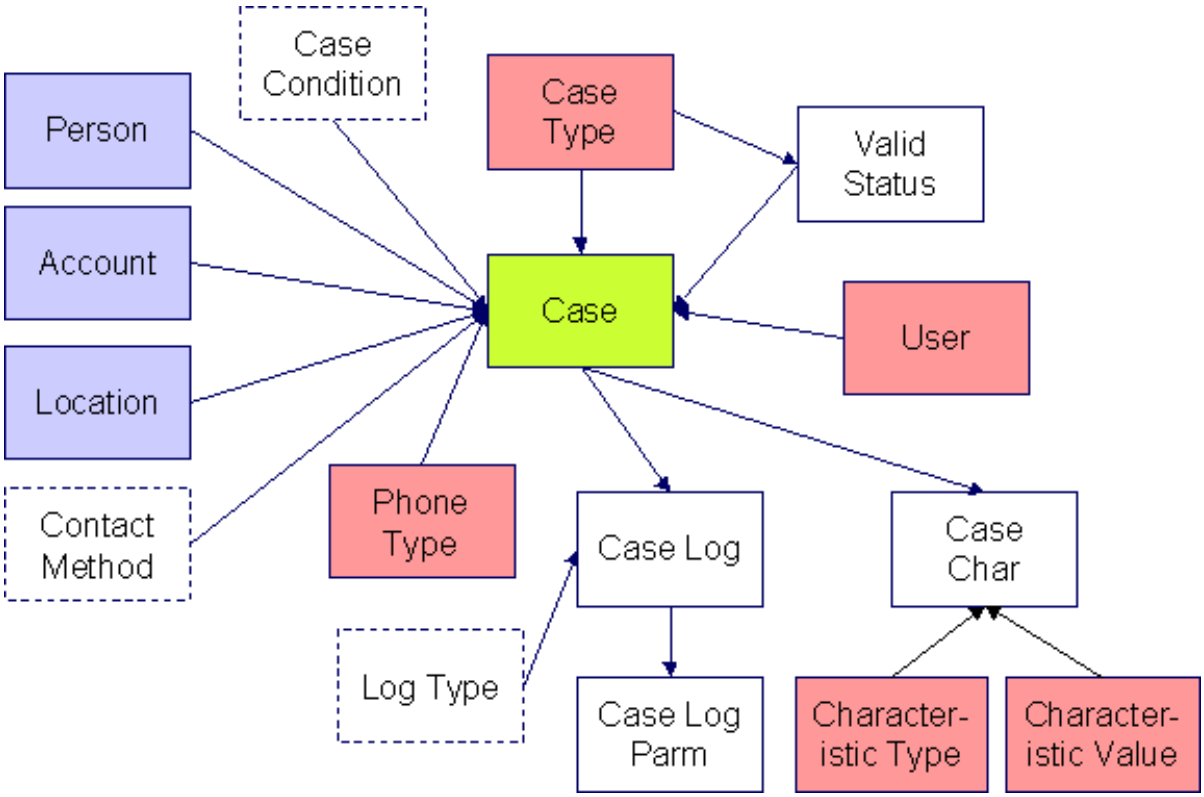
Collection Agency Referral Suggestions

N/A

Case

Case Data Model

The following data model illustrates the Case object.



Case Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Case	*CI_CASE*	Yes. *CI_CASE_K*	*VAL-CASE*		CIPVCSEK	CIPVCSEI
Case Characteristic	*CI_CASE_CHAR*	No. The key is case id, char type code and a sequence number		CIPVCCHV		CIPVCCHI
Case Log	*CI_CASE_LOG*	No. The key is case id and a log sequence number		CIPVCLGV		CIPVCLGI
Case Log Parameter	*CI_CASE_LOG_PARM*	No. The key is case id, log sequence number and a parameter sequence number		CIPVCPAV		CIPVCPAI

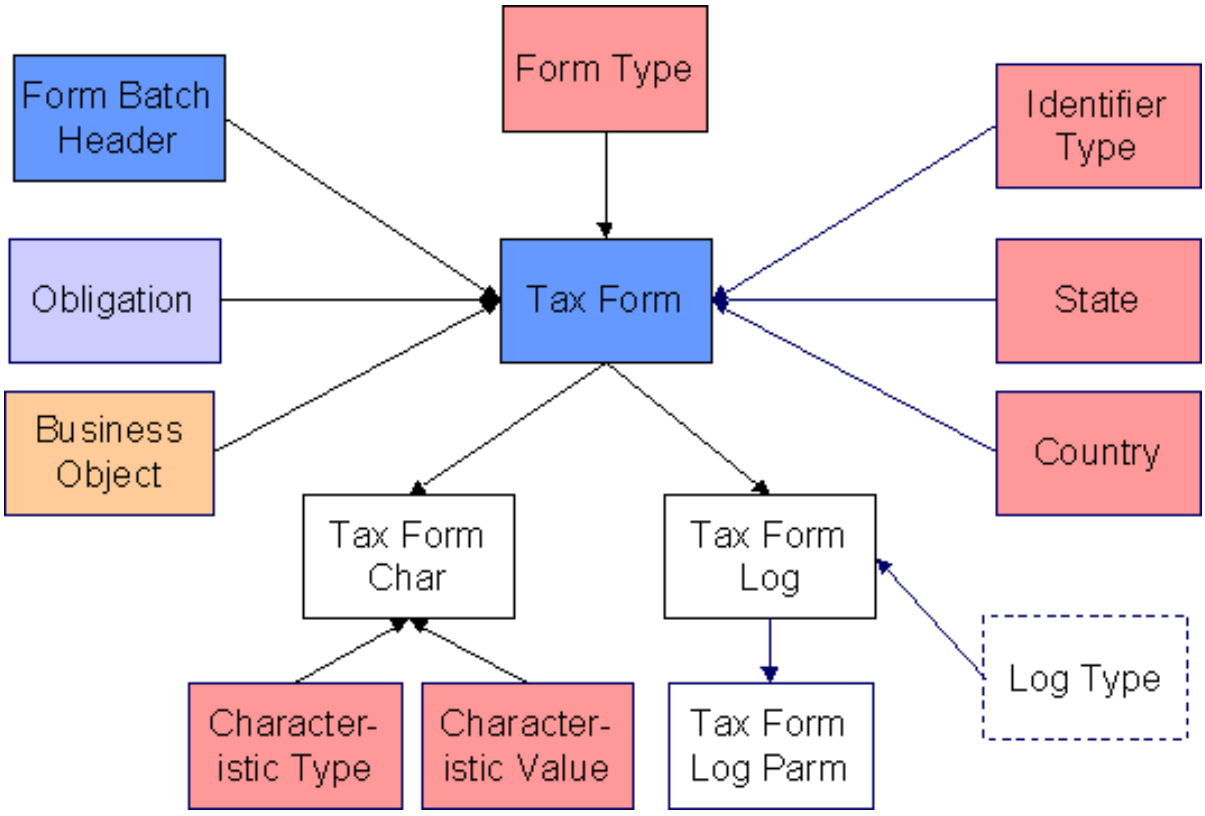
Case Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Tax Form

Tax Form Data Model

The following data model illustrates the Tax Form object.



Tax Form Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Tax Form	*CI_TAX_FORM*	Yes. *CI_TAX_FORM_K*	*VAL-TXFR*	CIPVTXFV	CIPVTXFK	CIPVTXFI
Tax Form Characteristic	*CI_TAX_FORM_CHAR*	No. The key is Tax Form ID, Characteristic		CIPVTFCV		CIPVTFCI

		Type, and Sequence.				
Tax Form Log	CI_TAX_FORM_LOG	No. The key is Tax Form ID and Sequence.		CIPVTFLV		CIPVTFLI
Tax Form Log Parameter	CI_TAX_FORM_LOG_PARM	No. The key is Tax Form ID, Sequence, and Message Parameter Sequence.		CIPVTLPV		CIPVTLPI

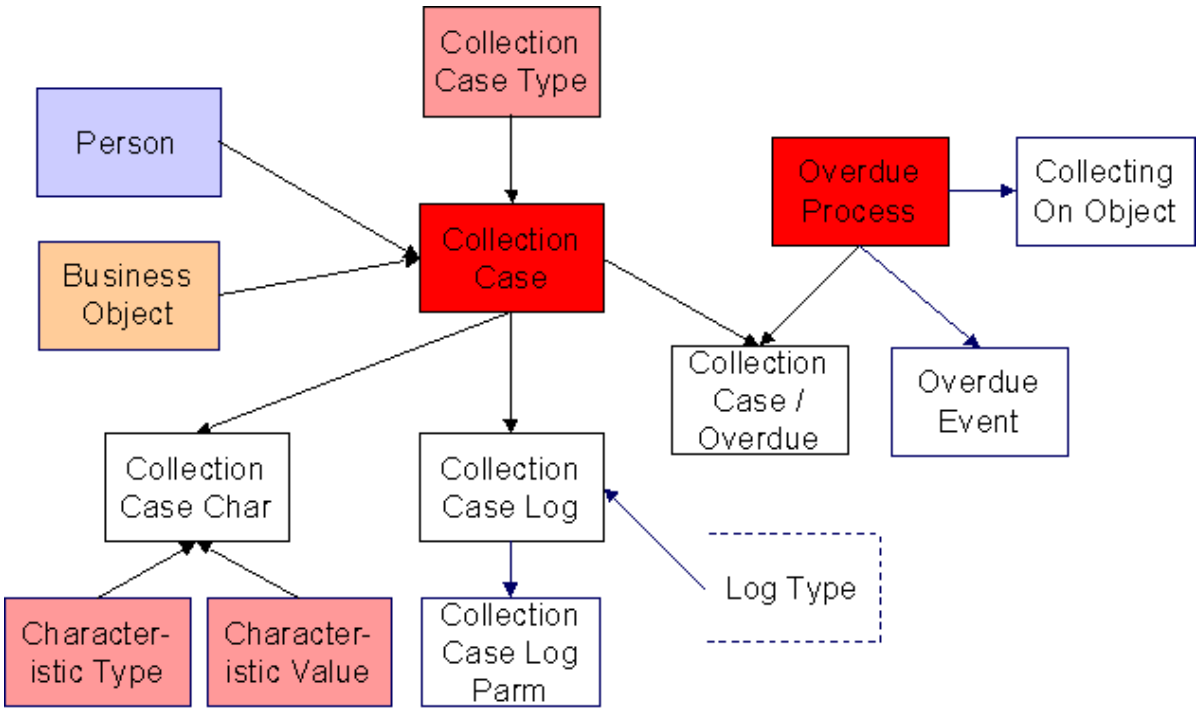
Tax Form Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Collection Case

Collection Case Data Model

The following data model illustrates the Collection Case object.



Collection Case Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Collection Case	*CI_COLL_CASE*	Yes. *CI_COLL_CASE_K*	*VAL-CLCS*	CIPVCCSV	CIPVCCSK	CIPVCCSI
Collection Case Characteristic	*CI_COLL_CASE_CHAR*	No. The key is Collection Case ID. Characteristic Type, and Sequence.		CIPVCCCV		CIPVCCCI
Collection Case Log	*CI_COLL_CASE_LOG*	No. The key is Collection Case ID and Sequence.		CIPVCCLV		CIPVCCLI
Collection Case Log Parameter	*CI_COLL_CASE_LOG_PARM*	No. The key is Collection Case ID, Sequence, and Message Parameter Sequence.		CIPVCCPV		CIPVCCPI
Collection Case Overdue Process	*CI_COLL_CASE_OD*	No. The key is Collection Case ID and Overdue Process ID.		CCIPVCCOV		CCIPVCCOI

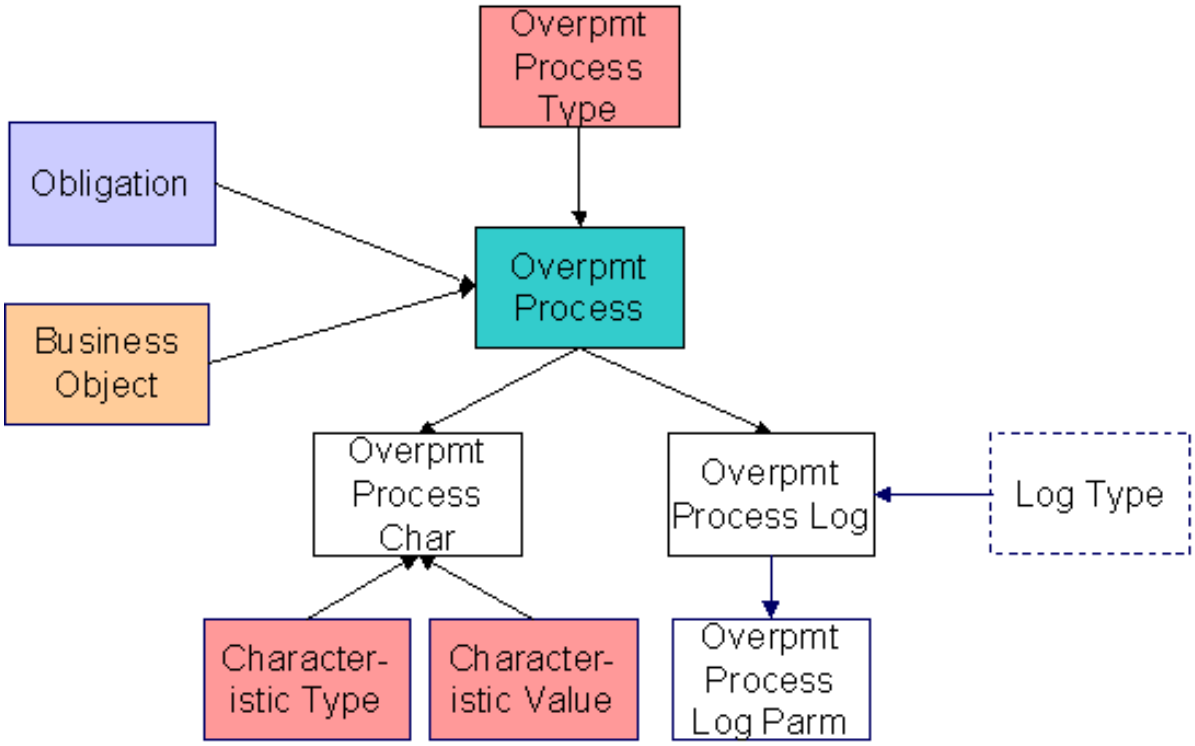
Collection Case Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Overpayment Process

Overpayment Process Data Model

The following data model illustrates the Overpayment Process object.



Overpayment Process Table Names

Data Model Name	*Table Name*	*Generated Keys*	*Object Validation Batch Control*	*Referential Integrity Validation Batch Control*	*Key Assignment Batch Control*	*Insertion Batch Control*
Overpayment Process	*CI_OP_PROC*	Yes. *CI_OP_PROC_K*	*VAL-OVPY*	CIPVOPPV	CIPVOPPK	CIPVOPPI
Overpayment Process Character	*CI_OP_PROC_CHAR*	No. The key is Overpayment Process ID, Characteristic Type, and Sequence.		CIPVOPCV		CIPVOPCI
Overpayment Process Log	*CI_OP_PROC_LOG*	No. The key is Overpayment Process ID and Sequence.		CIPVOPLV		CIPVOPLI
Overpayment Process Log Parameter	*CI_OP_PROC_LOG_PARM*	No. The key is Overpayment Process ID, Sequence, and Message Parameter Sequence.		CIPVOPMI		CIPVOPMI

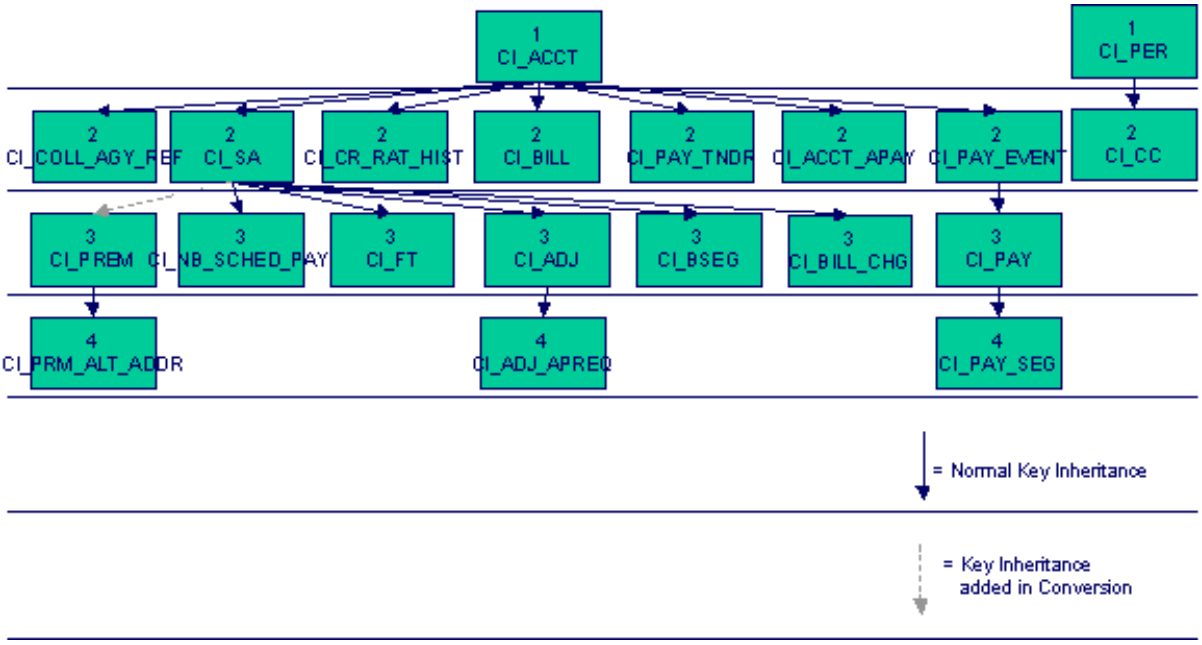
Overpayment Process Suggestions

This maintenance object includes a character large object field that your organization may be using to capture implementation specific data as defined by your business objects. For records of this type, the process to insert the records to the staging table is responsible for populating the data in this CLOB as per the record's business object schema.

Program Dependencies

The programs used to assign production keys are listed under *Master Data* and *Transaction Data* (in the Table Names matrices). Most of these programs have no dependencies (i.e., they can be executed in any order you please). The only exceptions to this statement are illustrated in the following diagram.

The tiers in this diagram contain a box for each table whose key assignment program is dependent on the successful execution of other key assignment programs. The numbers that appear in the boxes describe the order in which these programs must be executed.



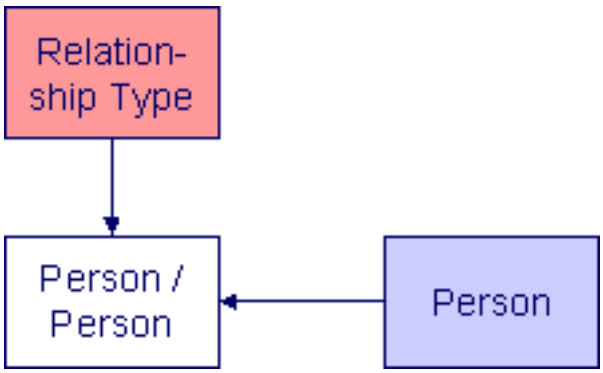
Please refer to the various "Table Names" sections above for the respective names of the programs to allocate each table's keys.

Caution:

Take note! Prior to running the key generation program for a particular object, it is required that any previously generated keys be cleared from the key allocation tables and the key allocation temporary storage table. It is recommended that the key allocation tables be analyzed between runs to maximize performance.

Appendix A - Entity Relationship Diagramming Standards

Because all data is stored in relational table, you need to be able to read diagrams that illustrate relationships between the various tables. The following entity diagram uses every diagramming notation used in this course:



Entity

Every box on the above diagram represents an entity (i.e., a table). An entity may be a physical entity, such as a Person, or a logical construct, such as an Account.

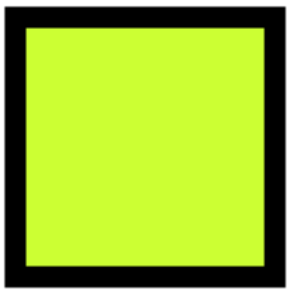
Color Coding

If you can view this document in color, you will notice that each entity is colored. The color indicates a logical grouping of entities. For example, form related entities are one color and taxpayer information related entities are a different color.

Some entities are not color-coded (i.e., they are white). These entities do not have a dedicated page, as they are part of a parent entity. For example, the Person / Person entity is related to the Person object and does not have its own maintenance object.

The following table describes the colors utilized in the documentation:

Color	***Grouping***
	Taxpayer Information
	Admin (Control) Table. These tables are referenced as foreign keys on master and transaction tables. We do not document the names of these tables in this document as the table names are easily accessible using the Table transaction.
	Child table - the entity is maintained in respect of a higher level entity.
	Lookup - the values in these types of entities are defined in a special table referred to as the lookup table. In order to determine the valid values for a column that references a lookup table, use the name of the column as the search value on the Lookup user interface.
	Metadata.

	Adjustment
	Financial Transaction
	Rates
	Billing
	Collection
	Overpayment
	Forms
	Case

Relationships

The solid line connecting the two entities that is terminated by an arrow represents a relationship between two entities. You read the relationship from the entity without the arrow to the entity with the arrow. For example, the line

between account type and Account illustrates that an account type may have many Accounts, but an Account may be part of a single account type.

Appendix B - Multiple Owners In A Single Database

In the schematic referenced in the *Introduction*, you'll notice that there are two table owners in the system database. We refer to the first owner as "staging" and the second owner as "production".

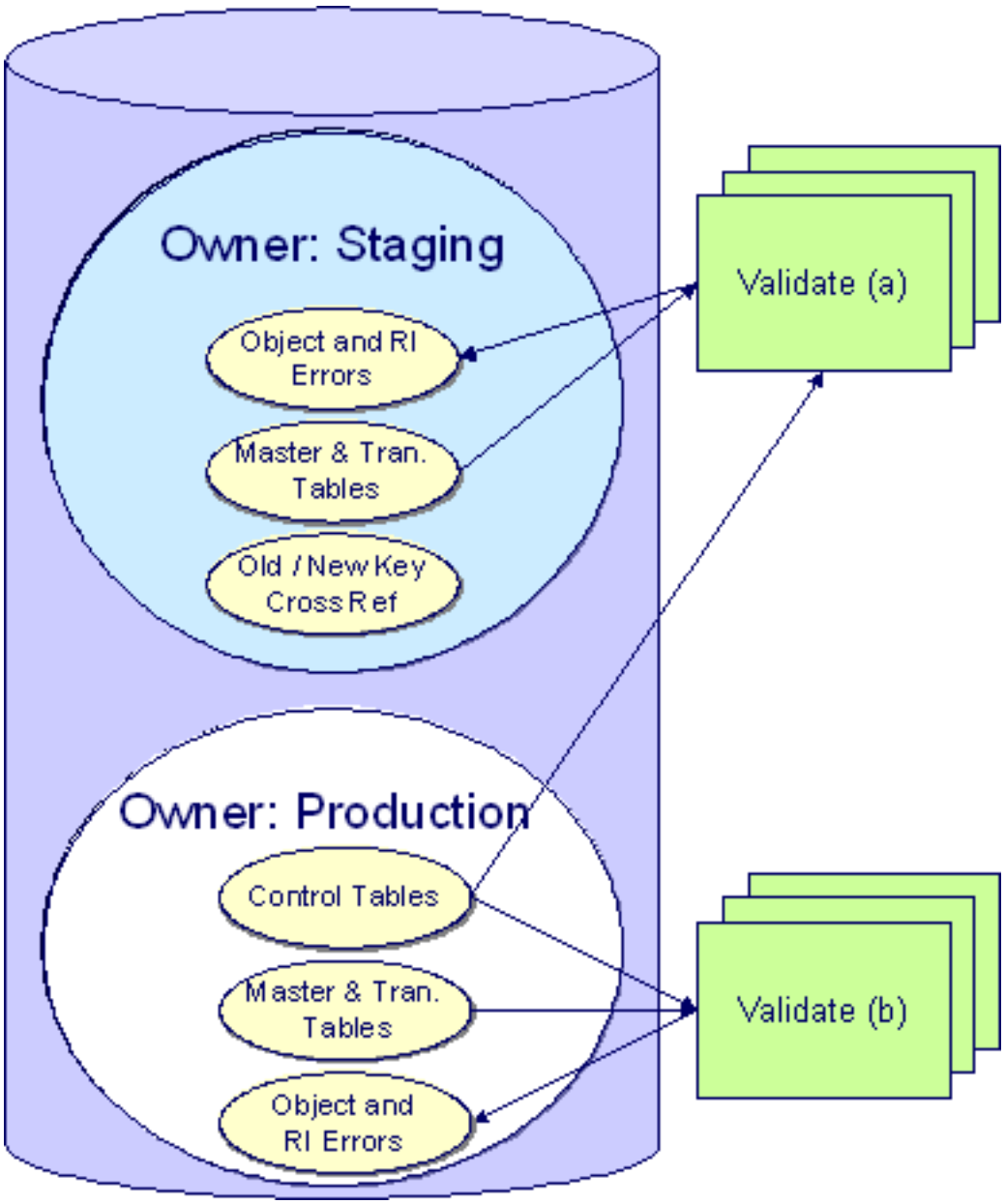
The staging owner is linked to the tables into which you insert your pre-validated data. These tables have an owner ID of **CISSTG**.

Note:

Multiple staging databases. It is possible to have multiple staging databases. In this situation, each one would have a unique owner ID, e.g., **CISSTG1**, **CISSTG2**, etc.

The production owner is linked to the tables used by your production system. These tables have an owner ID of **CISADM**.

When the validation programs run against your staging data, they validate the staging data against the production control tables (and insert errors into the staging error table). This means that the SQL statements that access / update master and transaction data need to use the staging owner (**CISSTG**). Whereas the SQL statements that access control tables need to use the production owner (**CISADM**).



But notice that when these same programs run against production (Validate (b)), the SQL statements will never access the staging owner. Rather, they all point at the production owner.

This is accomplished as follows:

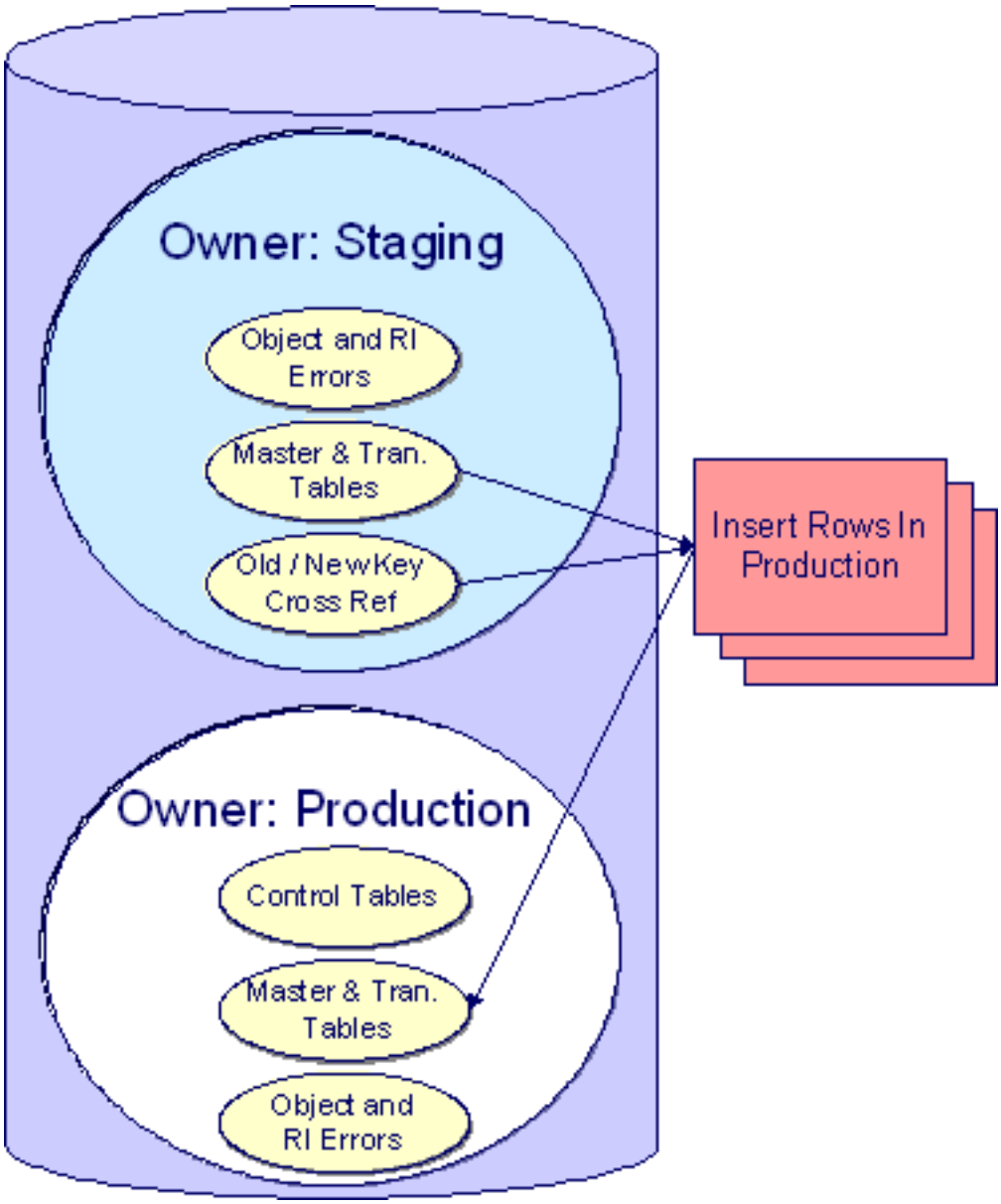
- A separate application server must exist for each owner. Each application server points at a specific Tuxedo server. The Tuxedo server references a specific database user ID.
- The database user ID associated with the staging database uses **CISSTG** as the owner for the master and transaction tables, but it uses **CISADM** as the owner of the production control tables.
- The database user ID associated with the production database uses **CISADM** as the owner for the master, transaction, and control tables.

You may wonder why we went to this trouble. There are several reasons:

- We wanted to re-use the validation logic that exists in the programs that validate your production data. In order to achieve this, these programs must sometimes point at the staging owner, and other times they must point at the production owner (and this must be transparent to the programs otherwise two sets of SQL would be necessary).
- We wanted to let you use the application to look at and correct staging data. This can be accomplished by creating an application server that points at your staging database with the ownership characteristics described above.

- We wanted the validation programs to be able to validate your production data (in addition to your staging data). Why would you want to validate production data if only clean data can be added to production? Consider the following scenarios:
- After an upgrade, you might want to validate your production data to ensure your pre-existing user-exit logic still works.
- You may want to conduct an experiment of the ramifications of changing your validation logic. To do this, you could make a temporary change to user exit logic (in production) and then run the validation jobs on a random sample basis.
- You forget to run a validation program before populating production and you want to see the damage. If you follow the instructions in this document, this should never happen. However, accidents happen. And if they do, at least there's a way to determine the ramifications.

While the redirection of owner ID's is a useful technique for the validation programs, it cannot be used by the key assignment and production insert programs? Why, because these programs have to access the same tables but with different owners. For example, the program that inserts rows into the person table must select rows from staging.Person and insert them into production.Person.



This is accomplished as follows:

- Views exist for each table that exists in both databases. These views have hard-coded the database owner **CISADM** (production). For example, there is a view called CX_PER that points at person table in production.
- The key assignment and insertion programs use these views whenever then need to access production data.

Appendix C - Known Oddities

Be aware that the following tables reference master data (e.g., persons, accounts). This means that if you look at them using a user ID that defaults ownership to the staging level, you will not be able to see the related master data (because the person / account doesn't exist in the staging owner's tables).

- Collection Agency. References a person.
- Tender Source. References a suspense obligation.