Siebel Incentive Compensation Management Configuration Guide

Version 7.5.3
December 2003

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404

Printed in the United States of America

Contents

1 Overview of Incentive Compensation Setup

This guide is designed for system administrators or high-level plan designers that need to access the core components of the Siebel Incentive Compensation Management databases in order to configure Siebel Incentive Compensation Management to fit the company's business practices. It covers many of the configuration and customization functions of Siebel Incentive Compensation Management and discusses some of the back-end processes that govern how Siebel Incentive Compensation Management operates.

The companion to this book, the *Siebel Incentive Compensation Management Administration Guide*, covers most of the essential functionality in Siebel Incentive Compensation Management for designing and processing incentive plans. It also provides a basic guide to navigating and working within Siebel Incentive Compensation Management. It is intended for plan administrators and other common users of Siebel Incentive Compensation Management as well as new users. If you are new to Siebel Incentive Compensation Management, you should start by reading *Siebel Incentive Compensation Management Administration Guide* before proceeding with this manual.

The first section covers the first steps in setting up a brand-new installation of Siebel Incentive Compensation Management. The user will learn how to set up the system administrator, how to create enterprise and operating units, and how to set up calendars for these units. It also discusses the creation of other user ID names and assigning access privileges to these users.

The second section covers all of the customization features of Siebel Incentive Compensation Management. Users will learn how to build customized profiles for transactions and performance records, extend the attributes of data records, modify user interface labels, and create templates used in credit rules and plan calculation formulas.

The third section gives users an overview of each of the processing services and explains how Siebel Incentive Compensation Management runs these services. It also covers the use of XML in importing and exporting data, how these XML files are structured, and how to properly convert files from another system into a format that Siebel Incentive Compensation Management can use.

Finally, the appendices provide several examples of template scripts used in context attributes, condition templates, and recipient templates. These sample scripts will prove useful to the experienced JavaScript user who wishes to create her own customized templates and attributes in Siebel Incentive Compensation Management.

It is recommended that all administrators and advanced Siebel Incentive Compensation Management users read all the chapters in this manual to gain a full understanding of how Siebel Incentive Compensation Management's core components operate. Users may then concentrate more deeply on those chapters that relate directly to their responsibilities or to the immediate task at hand.

The Siebel Incentive Compensation Management Model

This introductory chapter helps explain some of the concepts around which Siebel Incentive Compensation Management is constructed and should aid the advanced user in understanding how the various pieces of Siebel Incentive Compensation Management work together. In this discussion it is assumed that you has already read most or all of the *Siebel Incentive Compensation Management Administration Guide*; if you have not done so, you should read that manual first before proceeding with the discussion below.

Entities

Each data record in Siebel Incentive Compensation Management is generally referred to as an *entity*. An entity is any object in the database, such as an employee history record or calculation formula or rule set, that Siebel Incentive Compensation Management can use for a specific purpose. The entities that Siebel Incentive Compensation Management uses can be classified into four major types.

System Entities are the fundamental database objects in Siebel Incentive Compensation Management. They are objects that form the base for other entities in the Siebel Incentive Compensation Management database, and without which other entities cannot be created. This includes enterprise and operating units, the calendar, service types, users and roles, and locales. It also includes customization entities such as profiles and custom fields, templates, extended attributes, and context attributes. In general, system entity attributes cannot be customized; their fundamental structure and usage cannot be modified. They may, however, provide the means for customizing other entities throughout the database.

Incentive Entities are the objects that Siebel Incentive Compensation Management uses to process incentive plans and calculate payout results. These include credit rules and rule sets, calculation formulas and formula components, performance measures, and plans. Although individual incentive entities can be designed in any way a user desires, the basic structure and usage of an incentive entity cannot be customized.

Setup Entities refer to any other data objects that help set up a company's incentive plans, outside of the incentive entities. This includes participants, any participant setup entities such as jobs and channel segments, organizations, products, and hierarchies. Setup entities are usually referenced by incentive entities during plan execution; for example, a credit rule may refer to product data to determine credit distributions, and a formula may refer to employee data to determine payout results. Setup entities may be modified but for the most part remain constant over several processing periods. For example, a company's organization structure or product line will generally remain constant over several months or even years; occasionally organizations will be restructured and new products will be introduced, but these are not recurring, periodic events. Setup entities can be customized through *extended attributes*. Extended attributes do not alter the basic types of data, or attributes, that are normally tracked for a particular type of entity. Rather, as the name suggests, they are additional attributes that extend the types of data tracked for an entity.

Transaction Entities are data objects that are either imported or entered into Siebel Incentive Compensation Management on a periodic basis (such as transactions, credits, and goals) or calculated by the Siebel Incentive Compensation Management system on a periodic basis (such as credits, calculation results, and payout results). They are also referenced by the incentive entities during plan calculation. Unlike other entities, which are not period specific, transaction entities are strongly tied to one specific period. This period dependence helps Siebel Incentive Compensation Management determine, for example, when payouts should be calculated for specific credit records. With the exception of calculation and payout results, transaction entities can be fully customized through the use of profiles. Profiles are a group of custom fields (or attributes) that determine what basic data is tracked for transaction entities. Profiles are reusable across all transaction entities of the same type (a single transaction profile can be used by any transaction) and custom fields are reusable across all transaction entities of any type (a single custom field can be used by any transaction, credit record, or goal record).

Attributes

Throughout the above discussion of entities, we have repeatedly used the term *attribute*. In everyday usage, an attribute is anything that describes an object; *green* is an attribute of a leaf, *hot* is an attribute of a fire, and so on. Thus, an attribute of an entity is anything that describes that entity – more specifically, an attribute is a specific type of data associated with a data record (or entity). An employee record is an entity, and the employee's social security number is an attribute of that employee record.

All Siebel Incentive Compensation Management entities are set up with a standard set of attributes (these are often called *system attributes* and may also be called *original attributes* if the attributes can be modified). Incentive entities and system entities have attributes that cannot be customized. For these types of entities you cannot add to, modify, or remove their standard attribute set.

Setup entities have a standard set of attributes to which *extended attributes* can be added. Extended attributes are customized according to the needs of each client. They appear, from the end user's perspective, as ordinary data entry or drop-down fields. They are typically used when the standard attributes of an entity do not cover every crucial aspect of that entity, and additional attributes are needed to accommodate the company's compensation plans. Extended attributes are defined at the *operating unit* level (see "Context" on page 12 and Chapter 2, "Enterprise and Operating Units"), thus extended attributes defined within one operating unit do not extend to any other operating units.

Most transaction entities (transaction headers and lines, credits, and goals) also have a standard set of attributes to which *custom fields* can be added. On the surface, custom fields function very similarly to extended attributes, and from the user's perspective they work the same way. Custom fields, however, are not added directly to an entity type. Instead, custom fields are used to build *profiles*. There are four basic types of profiles (one for each of the above transaction entities) and any number of profiles of the same type may be created within an operating unit. Transaction header and transaction line profiles are then used to create *transaction types*, while credit and goal profiles are used to build *measures*. Each transaction record is associated with a transaction type, while each credit record and goal record is associated with a measure. This association is what Siebel Incentive Compensation Management uses to determine which custom fields go with each record. A credit record, for example, refers to a specific measure, which in turn refers to one specific credit profile, and that profile refers to one or more specific custom fields. This method of indirectly associating custom fields with specific entities allows several different types of transactions, credits, and goals to exist within the same operating unit.

Calculation result records and payout result records, which are also classed as transaction entities, do contain a small set of standard attributes, but do not have custom fields or profiles defined for them. Instead, the system automatically creates the required custom fields for each result record during plan calculation.

Context

All operations performed in the Siebel Incentive Compensation Management system are done within a specific *context*. This helps define the scope of an operation or of a data entity. Certain setup entities are system-owned: that is, their settings span an entire ICM installed instance. All other entities are defined within the context of an operating unit. This means that they are only applicable to that operating unit and are not shared or used by any other operating unit, even if those units fall under the same enterprise unit.

Context is an important concept for crediting and calculation functions; the context defines the scope of the process. Credits, for example, are generated within the context of a single transaction line event, of a specific calendar period, and of a specific operating unit. This automatically defines what data is available to the system for the purposes of creating those credits.

Periods and Versions

As a sort of sub-category of the context concept, all functions within Siebel Incentive Compensation Management are done in the context of a specific calendar period. A record is created in a period, and may be modified in another period. A transaction may be entered in a specific period. A service is run within a specific period. Whatever the function is, this period is called the entity's *effective start period*. It tells the system when that record, in its current form, was first valid in the system.

Along with this goes the concept of *versions*. When a record is created and marked with a certain effective start period, that period marks the record as the first version of that record. If the record is later modified, then the new effective start period becomes the period in which the edit was made, and this marks the beginning date of the second version of that record. At the same time that period also becomes the *effective end date* of the older version. This may continue infinitely, as records are continually modified or adjusted.

Versions are crucial to Siebel Incentive Compensation Management because they help keep records of every change or addition made to the database. They also allow a user to essentially "jump back in time" and review the system as it was within a prior period. Thus, if the user is currently working in period 6 and jumps back to period 1, then that user will see what the database looked like in period 1, not in period 6.

When multiple changes are made to a single record within a single period, the last change is always the one that Siebel Incentive Compensation Management considers valid for that period. When the Close Period service is run, the system captures the versions of each record in the database as they are at that moment, and those are the records that are preserved for future reference.

2 The System Administrator

The system administrator is the user that is responsible for the initial setup of Siebel Incentive Compensation Management and has the power to create enterprise units and assign Enterprise Unit administrators (see "Enterprise and Operating Units"). The system administrator may be the same person as one of these other administrators, but if this is the case then that person should use two or more user IDs, one for the system administrator role and a separate one for each other role.

The system administrator user ID and password must be setup before any other functions in Siebel Incentive Compensation Management can be used, even before the ICM application server is started because the system administrator is the only user that can set up enterprise units and operating unit and assign administrator users to those units.

Creating the system administrator user ID is a two-step process. First, the administrator must generate a secure password code for the ID. Second, the administrator must open the SiebelICMconfig.xml.in file and set up the ID using a chosen name and the secure password. The procedure below assumes that Siebel Incentive Compensation Management has been completely installed on one or more server systems.

Setting Up the Incentive Compensation Management System Administrator

1. If you have installed Siebel Incentive Compensation Management on a Windows NT server, open a new Command Prompt window by clicking Start, choosing Run, and typing cmd.
   If you are using a Solaris or AIX server system as the Siebel Incentive Compensation Management server, you need only log into this system to begin the process below.
2. Change your current directory to the Siebel Incentive Compensation Management main install directory.
3. Type generatepassword password.
   Replace the word "password" above with the actual password you want to use for the system administrator ID. The password may only contain letters and numbers.
4. The system will return a string of letters and numbers between a pair of square brackets ([ ]). This is the hash equivalent of the password you entered. Select this string, without the brackets, and copy it.

   NOTE: When the system administrator logs into the Siebel Incentive Compensation Management system he or she will still enter the chosen password. The system will automatically translate the entered password into its hash equivalent and compare that value to the password value found in the SiebelICMConfig.xml.in, and allow access only if those two values match. This prevents any other user from discovering the system administrator's password, because if a user attempts to enter the hash equivalent password value found in the properties file, the system will convert that value into its own hash equivalent and the resulting value will not match the value in the properties file, thus preventing a successful logon.

5 Open the SiebelICMConfig.xml.in found in the config directory of the Siebel Incentive Compensation Management installation, using a plain text editor.

6 Locate the following lines in this file:

```
system.admin.username=
system.admin.password=
```

CAUTION: If you change either of these lines with the SiebelICMConfig.xml.ini file, you must run setup jboss again.

7 In the username line, type in the user name you want the system administrator to use. This may be any text value. Do not include spaces or punctuation marks in the username.

8 In the password line, paste the hashed password string from step 4.

NOTE: Do not enter the actual password that you want to use for this username.

9 Save the SiebelICMConfig.xml.in, and close it.

10 You are now ready to use the system administrator user ID and log into Siebel Incentive Compensation Management to continue with initial setup.

Next Steps

Once you have successfully set up the system administrator user ID, you must perform the following initialization steps:

1 Log into the bootstrap directory of Siebel Incentive Compensation Management and set up an enterprise unit.

2 Set up one operating unit for the new enterprise unit.

3 Log into the Siebel Incentive Compensation Management application, using the enterprise unit administrator ID, and set up a calendar for the enterprise unit.

4 Log into the Siebel Incentive Compensation Management application again, using the operating unit administrator ID, and set up a calendar for the operating unit.

3 Enterprise and Operating Units

An enterprise unit is the basic building block for your compensation plans. The enterprise unit can be used to represent a company, whether it is a single entity or an extended enterprise that encompasses numerous companies. At least one enterprise unit must be set up in the system before any other data can be entered or imported into the system. Enterprise units cannot be set up through the normal Siebel Incentive Compensation Management interface, but must be set up through a special system administrator's interface.

Any number of enterprise units may be set up in Siebel Incentive Compensation Management, but the system will not treat these as related units; that is, data related to one enterprise unit is not shared with any other enterprise unit, plans related to an enterprise unit are unrelated to plans in other enterprise units, and so on.

Every enterprise unit must have at least one enterprise unit administrator, who has the ability to modify any data within the enterprise unit and its operating units and is the only user that may add other users to the system and assign appropriate security roles to them. Enterprise unit administrators are added when the enterprise unit is created; all other users and types of users are defined after the enterprise unit has been set up.

Setting Up An Enterprise Unit

The following task provides instruction on how to set up an enterprise unit.

To set up an enterprise unit

1. Open your preferred Web browser on your Siebel Incentive Compensation Management server and navigate to the following address: http://<app.host>:8080/Siebel/bootstrap.
2. Log into the application using the system administrator username and password.

   The Bootstrap screen will appear.
3. Click Create New EU.
4. Enter a unique identifying code for the enterprise unit in the Enterprise Unit Code field. Use the Enterprise Unit Description field to type in the full name of the enterprise unit.
5. Set the Begin Effective Date of the enterprise unit to the first day of the earliest calendar year that you will create in all of the operating units that this enterprise unit will contain.
6. Leave the Status field set to Active.
7. In the Functional Currency field select the currency in which the enterprise unit typically processes transactions and payouts.

8 In the EU Admin User Name field, enter a username for this enterprise unit's administrator. The user name may contain any combination of letters and numbers, but may not contain spaces (use the underscore _ instead) or any non-alphanumeric characters.

NOTE: The EU administrator may be the same person as the system administrator, but each administrator role must use a separate username and password.

9 Enter the EU administrator's password in the Password field, then re-enter the same password in the Confirm Password field. Like the username, the user's password may contain only alphanumeric characters or the underscore character, and may not contain spaces. Also, the password must be at least six characters long and contain both numbers and letters.

10 Click Save.

A logout message will display.

11 Click OK.

Operating Units

An operating unit is the second fundamental building block of compensation plans. An operating unit is used to represent a major division or operating branch of the company. For example, if the enterprise unit represents an extended enterprise corporation, each operating unit might represent one separate company that is part of that larger enterprise.

At least one operating unit must be defined for each enterprise unit, and typically there will be multiple operating units that report to the enterprise unit. However, it is also common for only one operating unit to report to a single enterprise unit; this will often be the case if the company needs to administer incentive plans for one division within the company. Operating units cannot be shared by multiple enterprise units; if an operating unit logically should report to more than one enterprise unit, then that operating unit must be set up separately within each enterprise unit, although the system will still treat them as separate operating units.

Operating units are only related by their common enterprise unit, and do not share any data with each other beyond the data stored at the enterprise unit and system level.

Like enterprise units, operating units have operating unit administrators. These are users who have the ability to modify any part of the operating unit and its data, and typically are responsible for maintaining plan details and participant records. Typically, operating unit administrators are *not* the same people as enterprise unit administrators, as enterprise unit administrators have privileges to create other operating units.

Setting Up the First Operating Unit

New Operating Units can only be created from the bootstrap URL. The steps to create a new operating unit are outlined below.

To set up the first operating unit

1. When the initial enterprise unit has been set up and you have logged into the bootstrap URL as the new EU administrator, the Create EU/OU page will display.
2. Enter a unique identifying code for the operating unit in the Operating Unit Code field. Use the Operating Unit Description field to type in the full name of the operating unit.
3. Enter the Beginning Effective Date for the operating unit.

   CAUTION: The beginning effective date of the operating unit will determine the earliest beginning date of the operating unit's calendar. Be sure to choose the effective date to correspond to the beginning of the company's or branch's calendar, *not* the date on which the operating unit is created.
4. Leave the Status field set to Active.
5. In the Functional Currency field select the currency in which the operating unit typically processes transactions and payouts.

   NOTE: The operating unit's functional currency may be the same currency type as for the enterprise unit, or it may be a different currency. When processing transactions, credits, and payouts, the operating unit's functional currency will override the enterprise unit's functional currency.
6. In the OU Admin User Name field, enter a username for this operating unit's administrator. The user name may contain any combination of letters and numbers, but may not contain spaces (use the underscore _ instead) or any non-alphanumeric characters.

   NOTE: The OU administrator may be the same person as the system administrator, but each administrator role should use a separate username and password.
7. Enter the OU administrator's password in the Password field, then re-enter the same password in the Confirm Password field. Like the username, the user's password may contain only alphanumeric characters or the underscore character, and may not contain spaces. Also, the password must be at least six characters long and contain both numbers and letters.
8. Click the Save button at the bottom of the screen.

   A message will display stating that the EU and OU have been created.
9. Click Login and the calendar creation page for your new OU will display.
   This completes the initial setup of the enterprise unit and operating unit.

Setting Up Additional Operating Units

Since all operating units are created in the same manner through the bootstrap URL, follow the instructions listed above to create additional operating units.

Import Employees

This imports new employee records and updates existing records, based on data in an external import file. The association of job codes to employees requires the job codes to exist in the system, thus, this import service should be run after the import job code service.

To associate an employee with a supervisor, the supervisor record must exist in the database. It is recommended that you create separate import files for each group of participants, (such as, employees, supervisors, managers, and executives) and import them from the top of the hierarchy down (such as Executives first, managers second, and so on.)

Operating Unit Level Data

Most of the data found in the Siebel Incentive Compensation Management system is stored at the operating unit level, meaning that data is usable only in the operating unit in which the data is created or imported. Such data includes:

- Products
- Employees, Channel Partners, and other participants
- Calendars
- Users and Roles
- Performance Measures
- Organizations and the organization hierarchy
- Territories and the territory hierarchy
- Plans and plan elements
- Calculation formulas and formula components
- Credit rules and rule sets
- Transaction data, including sales transactions, credit, calculation results, and pay outs

4 Calendars

The Siebel Incentive Compensation Management calendar defines the processing periods of your incentive plans. A Siebel Incentive Compensation Management calendar is composed of one or more calendar years. A *calendar year* in Siebel Incentive Compensation Management corresponds to a standard calendar year or fiscal year, as defined by your company's business practices. The *calendar* itself has no definitive end, as you can continually add more calendar years to the calendar, and thus the Siebel Incentive Compensation Management calendar is conceptually infinite in length.

The calendar is the first thing that must be set up after the system administrator has set up the first enterprise and operating units and assigned administrator user IDs to those units. When one of these administrators logs into Siebel Incentive Compensation Management for the first time, the system will detect that there is no calendar for the current operating unit and will automatically open the calendar year creation screen so that the administrator can initiate the calendar. This is also true any time a new operating unit is set up within an existing enterprise unit; when the operating unit administrator logs into that new operating unit for the first time, the system will prompt the user to create a calendar for that operating unit before permitting any other functions to be carried out.

Once the calendar has been initiated and one calendar year has been set up, future calendar years can be set up at any time by any user that has the proper privileges. Calendar years can be set up as necessary or they can be set up several months or years in advance.

Each calendar year is considered separate from other calendar years in the same calendar. Thus, it is possible to define a new calendar with a different number of periods than the prior calendar, or with customized segment types incorporated (see below).

Calendar Periods

Each calendar year is comprised of periods. Periods are defined by a start date and an end date, and contain all dates in between those two points. Whenever the system refers to specific dates, it will also look up the period that corresponds to those dates. For example, transaction lines have one or more events associated with them, and these events are marked by specific dates. Siebel Incentive Compensation Management matches these dates with the calendar year's periods to determine whether or not the event occurred in the current period, in a prior period, or in a period that has not yet been processed.

The number of periods in your calendar year will correspond to how frequently you need to process your incentive plans. Most plans are calculated on a monthly basis, so most calendar years will contain twelve periods. Many other plans are calculated quarterly (four periods), semi-annually (two periods), and annually (one period). You can also define custom calendar years, for which you can define the calendar to have any integral number of periods, up to 365 (one day per period).

Every period in a calendar year is assigned a *Relative* period number and an *Absolute* period number. The relative period number corresponds to the period number within the calendar year itself, while the absolute number corresponds to the period number in relation to all other period numbers in all other calendars. For example, suppose your calendar follows a regular twelve-period pattern, and you are setting up the third calendar year for this calendar. Each period in that calendar year will be assigned a *relative* period number from 1 through 12. Each will also be assigned an *absolute* number of 25 through 36; relative period 1 corresponds to absolute period 25, period 2 to absolute period 26, and so on.

Note that Siebel Incentive Compensation Management does not require every period in a calendar year to contain the same number of days; the clearest example of this is a monthly calendar, in which periods begin and end on the first and last days of each month rather than each period being defined as 30 days in length. This permits complete flexibility in defining your calendar year periods and the dates that correspond to them.

Calendar Segments

Every calendar year has one or more segment types associated with it. A segment is defined by a start period and an end period, and contains all periods in between. For example, a quarter segment would correspond to a set of three month-long periods, and every calendar year would contain four quarter segments. An annual segment is simply the set of all periods within the calendar year.

Segment types simply denote different kinds of segments. For example, the *quarter* segment type is defined so that a calendar year can be divided into four such quarter segments. The *annual* segment type is defined so that any calendar year contains only one such segment, spanning the entire year.

Any number of segment types may be associated with each calendar year. The period type of the calendar year automatically defines one segment type, the segment type of the calendar year's periods. Other segment types may then be added to the calendar year. For example, if a calendar is set up as monthly (twelve periods), then the segment type *Month* is automatically associated with that year. You could then associate the *quarter* segment type with the calendar year as well, each quarter containing three of those monthly periods. Additionally, you could add an *annual* segment type, which would consist of one segment that contains all twelve of those periods.

You may also define custom segment types. Typically custom segment types are created for calendars that do not follow the standard 365/366 pattern, but they may also be created within regular calendars. To define a custom segment you simply need to decide how many segments of that type will be fitted into a calendar year, and then assign that type a unique name. Thus, you could create a segment type *Trimester* that contains three segments for one year; when added to a regular calendar year, this would normally divide the year into three segments each containing four monthly periods.

Note that like periods, each segment in a calendar does not need to contain the same number of periods as all other segments. As an example, suppose you needed to divide your monthly calendar year into five segments. You could assign three periods to each of the first three segments, then two periods to the fourth segment, and one period to the fifth segment. This adds to the flexibility of calendars to accommodate any arrangement of plan calculation and incentive payout schedule.

Adding a New Calendar Year

The calendar is the first thing that must be set up after the system administrator has set up the first enterprise and operating units and assigned administrator user IDs to those units. You perform this task from the Calendar view.

To add a calendar year

1. Click Company on the Main Navigation bar, then select Calendar.
2. Click New Calendar Year.
3. Enter a code for the new year in the Code field.
4. Choose one of the following:
   - **Standard Calendar Year** – Select this if the year will follow a regular 365 or 366 day standard calendar, and all of the periods and segments in the year will be regular. Continue with step 5 under Standard Calendar Years.
   - **Custom Calendar Year** – Select this if the year does not follow a standard 365/366 day pattern and/or if you need to set up custom periods or segments. Continue with step 5 under Custom Calendar Years.

To set up standard calendar years

1. Select a Start Date for the calendar by choosing a month and year to begin the calendar. The first date of the calendar will be the first day of the chosen month.
2. Under Period Segment Type, choose your calendar's period type. This choice will also determine how many periods are contained in the calendar year.
3. Click Next to continue.
4. The calendar display shows each Relative Period number, its Absolute Period identity, and the associated date ranges with each period. You may modify the Start Date of any period except the first, and the End Date of any period except the last. Note that the end date of a period must always be one day prior to the start date of the next period.
5. Click the Save button to continue.

   The Segments screen shows the different segments that have been automatically created for your calendar. For regular calendars, the chosen period type is automatically considered a segment. Any segments that are longer than the period segment type will be automatically generated by the system. You may click Save again if your calendar does not require custom segment types to be set up, and skip the remaining steps. Continue with step 10 to create custom segments.
6. If custom segment types have already been defined in other calendar years, you may choose that segment type under Custom Segment Types, then click the Add button.

   To create a new segment type for this calendar:

   - Select *New Custom Segment Type*, then click the Add button.

- In the Segments Per Year field, enter the total number of periods that will appear in the calendar year.
- Enter an identifying code for this segment in the Segment Type Code field.
- Click Next. This will return you to the Segments screen. Under Custom Segment Types, select your new segment type and click the Add button.

7 After clicking the Add button, the system will automatically try to fit the chosen segment type into the calendar year and distribute periods to each segment.

8 The Define Segments section displays the number of segments for this segment type, and the calendar periods that will be associated with those segments. You may change the First Period for any segment except the first, and the Last Period for any segment except the first.

9 Click the Save button to complete adding the segment to the calendar.

10 Repeat steps 10 through 12 for each custom segment type you want added to the calendar.

11 Click Save to complete the calendar.

To set up custom calendar years

1 Select a Start Date for the calendar by choosing a month, date, and year to begin the calendar.

2 Select the calendar's End Date by choosing the last date, month, and year of the calendar.

3 Under Period Segment Type, you may select any existing custom segment type to define the number of periods in the custom year.
If no custom segments have previously been defined for other calendar years, you may only select *New Custom Segment Type* to define the calendar's segments. Follow this procedure:

- Click Next to proceed.
- In the Segments Per Year field, enter the total number of periods that will appear in the calendar year.
- Enter an identifying code for this segment in the Segment Type Code field.
- Click Next. This will return you to the first calendar screen in step 3 above. You must then redo steps 5 through 7 to set up the custom calendar, although you may now choose the correct Periods Per Year.

4 Click Next to continue.

5 The calendar display shows each Relative Period number, its Absolute Period identity, and the associated date ranges with each period. Note that Siebel Incentive Compensation Management automatically tries to calculate an even distribution of days amongst the chosen number of periods.

You may modify the Start Date of any period except the first, and the End Date of any period except the last. Note that the end date of a period must always be one day prior to the start date of the next period.

6 Click the Next button to complete the calendar year and continue with custom segment definition.

7 In the Custom Segment Types field, select an existing custom segment type code, then click Add. If you need to add a new segment type:

- Select New Custom Segment Type and then click Add.
- In the Segments Per Year field, enter the total number of periods that will appear in the calendar year.
- Enter an identifying code for this segment in the Segment Type Code field.
- Click Next. This will return you to the calendar screen for step 11. You may now add the custom segment type.

8 The Define Segments section displays the number of segments for this segment type, and the calendar periods that will be associated with those segments. You may change the First Period for any segment except the first, and the Last Period for any segment except the first.

9 Click Save to add the segment type.

- Repeat steps 11 through 13 to continue adding segment types to the calendar.

10 Click Save to complete the calendar.

Combining Custom Calendars and Standard Calendars

In many situations it will be necessary to mix standard calendars and custom calendars either in the same calendar year or across calendar years. If the number of periods in a calendar year changes from one year to the next, formula Pay When options and Draw Recovery period options will be affected. For more information, see *Siebel Incentive Compensation Management Administration Guide*.

EXAMPLE – Semi-Annual Bonus

FY2002 has 12 segments in the calendar matching a standard monthly calendar. A bonus program is in effect to be paid in periods 6 and 12 (effectively June 2002 and December 2002). FY2003 has been updated to reflect 24 segments in the calendar beginning January 2003 and ending December 2003. The formulas designed to calculate the semi-annual bonuses will need to be adjusted to reflect the new Pay When period(s). Unadjusted formulas will pay this bonus in March 2003 and June 2003 (effectively periods 6 and 12 in the new calendar).

EXAMPLE – Draw Recovery

FY2002 has 12 segments in the calendar matching a standard monthly calendar. A draw program for all new participants is in place that pays a $2000 draw in periods 1-3 and begins draw recovery in period 4 (Effectively April 2002). FY2003 has been updated to reflect 24 segments in the calendar beginning January 2003 and ending December 2003. The draw recovery settings will need to be adjusted to reflect the new segments in the calendar. Unadjusted draw recovery will begin draw recovery in February 2003 (Effectively period 4 in the new calendar).

5 Security Settings

This chapter details security settings in Siebel Incentive Compensation Management. The tasks included are as follows:

Roles in Incentive Compensation Management

A role is a set of privileges that specify which areas of the Siebel enterprise can be accessed by a user, along with the specific access rights to those areas. Every user must be assigned one or more roles in order to gain access to Siebel Incentive Compensation Management.

Some roles, such as *Enterprise Unit Administrator* and *Operating Unit Administrator*, are already defined in Siebel Incentive Compensation Management as part of the initial configuration. These roles have privileges already defined for them, and all the user needs to do is assign such roles to the appropriate users. In addition, you may define new roles to fit the security policies for your system and database.

Privilege Types

There are several types of privileges that define specific access rights to various parts of the Siebel Incentive Compensation Management system:

- **Create**. This privilege allows a user to create new records of the specified category, such as new products or new employees. For most categories of privileges, the privilege to create is combined with the Edit privilege.
- **Edit**. This allows a user to edit existing records of the specified category; this includes the privilege to delete existing records. For most categories of privileges, the ability to edit records is combined with the Create privilege.
- **Search**. This allows users to perform search queries on the specified category of data, such as a search for a product or a search for a formula. The Search privilege, when available, is always combined with the View privilege.
- **View**. This allows a user to open records for review, but does not allow editing or creating. The View privilege is almost always combined with the Search privilege for each privilege category.
- **Assign**. This type of privilege applies to assigning users to specific roles. For example, the system administrator typically has the right to assign other users to the Enterprise Unit Administrator role or to the System Administrator role.

- **Revoke**. This privilege specifically applies to roles and users. It permits the user to remove assigned roles from other users, similar to the ability to assign roles to users. Revocation of roles is usually done only when users change job functions.
- **Adjust**. This type of privilege applies to transactions, credit records, and payout records. It allows the user to enter adjustments to these types of records. This includes regular adjustments, returns, reversals, cancellations, and so on.

Some privilege categories, such as processes and analytic functions, do not have a privilege type. Assigning access to a process or an analytics function means the user can run that process or access that particular reporting function.

Privilege Categories

Each privilege category encompasses a specific function within Siebel Incentive Compensation Management, such as *Employees* and *Formulas*. For most categories you may assign Create/Edit privileges to the role and/or the Search/View privilege. By assigning such privileges to a role, you give those privileges to any user that is assigned to that role.

Following is a list of the major privilege categories along with their subcategories:

- System Administrator
  - Assign Enterprise Unit Administrator
  - Assign System Administrator
- Enterprise Unit Administrator
  - Assign Operating Unit Administrator
  - Assign Enterprise Unit Administrator
- Unit Setup
  - Incentive Types
- Transaction Setup
  - Profiles
  - Transaction Line Types
  - Transaction Event Types
  - Transaction Types
  - Custom Fields
- Templates
  - Condition Templates
  - Credit Recipient Templates
- Labels and Attributes
  - Extended Attributes

- Context Attributes
- Labels

- Company
  - Channel Segments
  - Payroll Systems
  - Locations
  - Jobs
  - Rate Groups
  - Cost Centers
  - Salary Grades
- Users and Roles
- Currency
- Hierarchies
  - Level Sets
  - Organizations
  - Products
  - Territories
  - Hierarchies
- Participants
  - Employees
  - Channel Partners
  - Customers
- Fundamentals
  - Jobs
  - Salary Grades
  - Payroll Systems
  - Cost Centers
  - Locations
  - Channel Segments
- Performance
  - Measures and Goals
  - Credits
- Plan Elements

- Draw Profiles
- Cap Profiles
- Formulas
- Plans
- Rounding Rules
- Expressions

- Transactions
  - Transactions
  - Credit Rules & Rule Sets
- Payout
- Services
  - Run
  - Import
  - Export
- Dashboards
  - Dashboards

Adding New Roles

Periodically you will need to add new roles to your system to allow new and existing users to gain access to Incentive Compensation Management.

To add new rolls

1. Click Company on the Main Navigation bar, then click Roles.
2. Click the New Role button.
3. Enter a unique name for the role in the Name field.
4. If this role is being set up for operating unit administrators, select the Operating Unit Admin check box. This will automatically assign all operating unit privileges to the role. You may continue with editing privileges (see below) or you may leave the privilege assignments as they stand.
   Note that this check box only appears if you, the user, have the privilege in your own role to assign operating unit administrator roles. The chief system administrator will have this privilege, but other users may not.

5 As a shortcut to setting up the role, you may copy privileges from another role. Click the Copy Privileges from Role drop-down list and select the appropriate role.
Once you have copied privileges, you can continue with this procedure to add or remove privileges from the new role, or you may skip through the remaining screens without making any changes.

6 Click Next to continue.

7 Click the Edit button next to any privilege category to assign or change privileges to the role. The Performance Privileges screen is shown below; other category screens are similar.

8 Select any check box to assign that privilege to the role, or de-select a check box to remove that privilege from the role.

- To assign all of the privileges within a sub-category, click the All check box for that category.
- Note that in most cases, if you assign Create/Edit privileges to a role you will also need to assign Search/View privileges for the same category as well; otherwise, the user will not be able to search for or review records for editing.
- Click Save to return to the View Role screen.

9 Use the navigation links at the top of the page to return to the search screen, or click New Role to start the process again for a new role.

Users in Incentive Compensation Management

A user is any person who must access one or more functions in Siebel Incentive Compensation Management. Users are generally administrators (employees that are given access to create, edit, and review data records and modify incentive plans) or participants (employees, channel partners, or customers that are given access to specific reports and analyses). Users are system entities and gain access to a specific operating unit by the assignment of a role in that operating unit. The operating unit administrators in all operating units can view all user records, even if those users do not have privileges to access the data in those operating units. A single user may be given access to several operating units by being assigned a role by the different operating unit administrators. This feature allows a single user to access several different operating units with a single login name and password.

Note that not every participant must necessarily be made a user; in some cases, your incentive plan may call for paying participants that should not have access to any part of the system. Similarly, in some cases your administrators may also be plan participants.

A user is identified by a unique user name and is assigned a password. The user name and password must be entered when the user logs into Siebel Incentive Compensation Management.

One user, the generic System Administrator, is already set up with every Siebel Incentive Compensation Management installation. This allows the system administrator access to the system during initial setup. It is highly recommended that the system administrator change the settings of this user before adding any other users to the system; this user identity should be assigned a unique password that only the system administrator knows, in order to prevent other users from accidentally gaining access to the system through this user name.

Assigning Roles To Users

Every user is assigned one or more roles. Each role defines what parts of the Siebel Incentive Compensation Management system the user can access, and whether the user can only view records or create and edit records as well.

Any number of users can share the same role, and a user may be assigned to any number of roles. Assigning multiple roles to a user gives that user all of the permitted privileges in each role; if one role allows a privilege that another does not, then the role that allows the privilege takes precedence.

Adding new users and assigning rolls

1. Click Company on the Main Navigation bar, then click Users.
2. Click the New User button.
3. Your new user may either be a Participant User or a Non-participant User:
   - If the user has already been set up as a participant, select the appropriate Type (employee, channel, or customer) and then enter the specific participant's code in the Participant ID field. You can also search for the participant's code by clicking the button.
   - If the new user is a non-participant, enter that user's actual name in the appropriate fields. You may also enter the user's e-mail address.
4. Click Next to continue.
5. Enter the user's User Name.

   This is a unique identifier for the user. The user name may contain any combination of letters and numbers, but may not contain spaces (use the underscore _ instead) or any non-alphanumeric characters.
6. Enter the user's password in the Password field, then re-enter the password in the Retype Password field.

   Like the user name, the user's password may contain only alphanumeric characters or the underscore character, and may not contain spaces. Also, the password must be at least six characters long and contain both numbers and letters.
7. Verify that the new user's Status is set to Active.

   Any users that are put on Inactive status will not be able to access the system.
8. Click Next to continue.
9. Assign a role to this user by selecting the role's name in the Add Roles drop-down list, then clicking the Add Role button.

   Repeat this step if you want to assign more than one role to this user.
10. Click the Finish button to save the user and review it.

6 Currencies

Siebel Incentive Compensation Management is designed to handle transactions in multiple currencies. Normally Siebel Incentive Compensation Management processes transactions, credits, calculations and payouts in U.S. Dollars. Any of these values can be converted into the equivalent values in other currencies. This allows, for example, a transaction made in Canadian Dollars to be credited and paid upon in U.S. Dollars, or vice versa.

Currency Codes

You must create currency codes and then assign them to each operating unit.

To create currency codes

1. Click Company on the Main Navigation bar, then click Currency Codes.
2. Click Add Currency Code.
3. Enter the identifying code for the field in the Code field.

   NOTE: You should use the global three-character currency codes when entering new currencies. This code is usually the same as the ISO 4217 standard. (The ISO -- or International Organization for Standardization -- is a worldwide federation of national standards bodies.) In most cases, the currency code is composed of the country's two-character internet country code plus an extra character to denote the currency unit. For example, the code for Canadian Dollars is simply Canada's two-character internet code ("CA") plus a one-character currency designator ("D"). Lists of ISO currency codes are available on the web.
4. Use the Name field to enter the name of the currency.
   Example: Canadian Dollar, US Dollar
5. Enter the ISO code for this currency in the Numeric Code field.
6. Enter a description as necessary for this currency code in the Description field.
7. Enter the fraction name in the Fraction Name field.
   For US Dollars this value is Cents.
8. Enter the symbol for the currency in the Symbol field.
   For US Dollars this is $.
9. Enter the fraction symbol in the Fraction Symbol field.
   For US Dollars this is c.
10. Enter the ration of the whole to the fraction in the Ration of Whole to Fraction field.
    For US Dollars this value is 100.
11. Select the ISO Currency check box if this currency code is an ISO Currency.
12. Enter the ISO version in the ISO Version field.

13 Click Finish.

To set up currency conversion rates, you must assign currencies to each operating unit; only the assigned currency types will be valid for that operating unit. Once this setup has been completed, you can import daily currency exchange rates through the Service Manager. These conversion rates can be viewed through the Daily Conversion Rates function, found on the Company menu.

Assigning Unit Currencies

When assigning unit currencies, be sure you are working in the operating unit for which you want to set up unit currencies.

To assign unit currencies

1 Click Company on the Main Navigation bar, then select Unit Currencies.

2 Select the currency code from the drop-down list.

3 Click the Add Currency button.

4 Repeat steps 2 and 3 above to add more currencies to this operating unit.

Changing Currency Conversion Options

Siebel Incentive Compensation Management's currency conversion options affect how the system converts transaction currency amounts into the functional currency equivalents for the purposes of generating credit record currency amounts.

To change currency conversion options

1 Click Company on the Main Navigation bar, then select Currency Conversion Options.

2 Click the Edit icon to modify existing conversion options.

3 Select or de-select the Use Transaction Conversion Rate check box if necessary.

- When selected, the system will always attempt to use currency conversion rates from transaction headers to convert transaction currency amounts into their functional currency equivalents when generating credit records. If such a conversion rate is not found on a transaction then the system's conversion rates will be used instead.
- If this box is not selected, transaction amounts will be converted using the system's currency conversion rates to generate credit amounts.

4 Under Credit Currency Conversion Date, select one of the following options to determine how the system will decide which date's conversion rates to use when converting transaction amounts into functional amounts for credit records.

- **Daily Transaction Header** – The daily system rates are used, and the specific date is determined by each transaction header's date.

- **Daily Transaction Line Date** – The daily system rates are used, and the specific date is determined by each transaction line's date.
- **Daily Transaction Event Date** – The daily system rates are used, and the specific date is determined by the event date on each transaction line.
- **Period Transaction Header** – The period system rates are used, and the specific date is determined by each transaction header's date.
- **Period Transaction Line Date** – The period system rates are used, and the specific date is determined by each transaction line's date.
- **Period Transaction Event Date** – The period system rates are used, and the specific date is determined by the event date on each transaction line.

5 Click the Save button to save your modifications.

7 Custom Fields

About Custom Fields

Custom fields are the data fields on transaction headers, transaction lines, credit records, and goal records that are specially adapted to your company and its incentive plans. Custom fields are used to build profiles, which in turn are the basis for transaction types and for measures, and ultimately these are used to create actual transaction records and performance records.

Custom fields can be reused across multiple profiles of the same type or of different types; you do not need to define separate custom fields for each profile you want to create. This allows you to define a certain type of data and use that data type in both transaction records and performance records. For example, many transaction records will require a data field to hold the total sales amount of the transaction, and this could be set up as a custom field called *Sales_Amt*. This same field can be used in credit profiles as well, so that the sales amount on a transaction clearly translates to the sales amount on a credit record. This custom field could also be used in goal profiles, so that there is a direct link between the goal set for sales and the actual sales made.

Custom fields should not be confused with extended attributes. From a user interface perspective, they appear to perform the same function. Extended attributes, however, are extensions to *setup entities*, such as employees and organizations and any other entity that , such as employees and organizations, but do not apply to transaction or performance records. Custom attributes are usually optional and do not need to be set up for every installation. Custom fields, on the other hand, apply exclusively to transactions and performance records and *must* be set up in every Siebel Incentive Compensation Management installation. Without custom fields, transactions and performance records will not be able to record any useful information beyond their identifying codes.

Data Types

There are five data types that define what sort of data a custom field can hold.

- **String**. String data is treated as text, whether the data field contains letters or numbers or other acceptable characters (such as underscores). Although a string may only contain numbers, the system will still treat the data as a text string.
- **Number**. These data fields may only contain numbers, as the system treats the data as numbers and not merely text. This is an important distinction when dealing with mathematical operations, because the system will perform the operations differently according to the data type.
- **Currency**. This is a special type of number, indicating that the number should be formatted to match the system's functional currency. For example, for U.S. dollars the number will always contain two decimal places.
- **Date**. These fields store dates only. Dates are stored in MM/DD/YYYY format (for example, 01/10/2001).

- **Boolean**. Boolean fields handle *Yes/No* or *True/False* conditions. Typically, these types of fields appear on the interface as check boxes. Thus, if the check box is selected, the data field will read *True*, and if not selected it will read *False*.

Aggregate Operators

Aggregate operators tell the system how to cumulate and roll the data it finds in custom fields when the Cumulation Service or the Rollup Service are run. (See *Siebel Incentive Compensation Management Administration Guide*.)

If a measure is set up to cumulate data across periods or roll up data along the organization or territory hierarchy, the system refers to the measure's credit and/or goal profiles and looks up their associated custom fields, then checks the aggregate operator for each of those custom fields and performs the cumulation or rollup accordingly. If no aggregate operator is chosen for a custom field, then the system will not cumulate or rollup data for this field regardless of a measure's cumulation settings.

The available operators are:

- **SUM**. This indicates that all values for this custom field type are to be added together. This operator is available for custom fields with data types of Currency, Number, or Boolean.
- **MIN/MAX**. These operators find the minimum and maximum values amongst the available values, respectively. These operators are useful for numeric and currency fields for finding the largest or smallest value in a range of records. It can also be used for date fields. For date fields, the MIN function will find the record with the earliest date (MAX finds the latest date). This operator is available for custom fields with data types of Currency, Number, Date, or Boolean.
- **COUNT**. This operator simply counts how many records containing this data field there are in the system. It does not regard the actual values in the data fields, it simply retrieves the number of records. This operator can be used with any data type.

Only one aggregate operator may be associated with each custom field. If you find that you need to associate more than one operator with a custom field, you must create two similar custom fields, each with its own operator. For example, suppose you need to cumulate sales amounts for credit records, and you need both the sum total of sales amounts for all records and the max sales amount for all records. You would need to create two distinct custom fields, such as *Sales_Amt_SUM* and *Sales_Amt_MAX*, and associate the appropriate operators with each. You would also have to include both custom fields in the credit profile, and set up credit rule distributions to copy the same value from transaction lines to each field.

Reversible Fields

Some custom fields may be marked as Reversible. This setting applies only to custom fields that will be used in credit profiles, and indicates that any numeric or currency values stored in that field can be reversed in a credit adjustment. Fields that are not set as reversible cannot be reversed during credit adjustment.

Returnable Fields

Some custom fields may be marked as Returnable. This setting applies only to custom fields that will be used in transaction line profiles, and indicates that any numeric or currency values stored in that field can be adjusted automatically when a transaction line is returned. Fields that are not set as returnable will not be adjusted when transaction lines are returned.

Defining Custom Fields

You define custom fields in the Custom Fields screen.

To define custom fields

1. Click Configure on the Main Navigation bar, then click Custom Fields.
2. Click the New Custom Field button.
3. Enter an identifying code for the field in the Code field.

   This code uniquely identifies the custom field within Siebel Incentive Compensation Management.
4. Under Data Type, select the type of data that may be entered in this custom field.

   This automatically restricts what data the user can enter in a field. Thus, if the user enters an invalid value, the system will prompt the user to correct the entry.
5. Under Aggregate Operator, select an operator that will be used to cumulate and roll up data for this custom field.

   If this is a non-cumulative data field or you will not be using the rollup feature, leave this field marked as None.
6. If this custom field will be used in transaction profiles, you may select the Returnable check box to indicate that this field can be automatically returned.
7. If this field will be used in credit profiles, you may select the Reversible check box to indicate that this field can be reversed through credit adjustments.
8. Enter a description of how this field will be used in the Description field.
9. Click the Finish button to complete the custom field and review it.

8 Profiles

A profile is a set of custom fields. Profiles are referenced by transaction header and line types and by measures. These types and measures are associated with specific transaction and performance records, thus the associated profiles determine which custom fields appear in those records.

For example, suppose that the credit profile *Sales Credits* contains three custom fields: *Sales_Amount, Quantity*, and *Net_Attainment*. A *Direct Sales* measure is then set up that references the *Sales Credits* profile. Any credit record that references the Direct Sales measure is consequently associated with the *Sales Credits* profile. Thus, in addition to the standard data fields for all credit records (such as the associated measure), these particular credit records will contain the three custom fields listed above.

As another example, suppose that a transaction line profile *Standard Sales Detail* is created and contains four custom fields: *Sales_Amount, Quantity, Profit_Margin*, and *Net_Profit_Amount*. A transaction line type *Standard Sale* is then set up that references the *Standard Sales Detail* profile. Any transaction line that references the Standard Sale line type is consequently associated with the Standard Sales Detail profile. Thus, in addition to all the standard data fields for all transaction lines (such as line ID and entry date), these particular transaction lines will contain the four custom fields listed above.

There are four types of profiles: transaction (header) profiles, transaction line profiles, credit profiles, and goal profiles. Any number of profiles may include the same custom field, so that transaction line profiles may contain the same fields as credit profiles, credit profiles may share common fields with goal profiles, and so on. All profiles are defined according to the process outlined below.

Setting up Profiles

You set up profiles in the Profiles screen.

To set up profiles

1. Click Configure on the Main Navigation bar, then click Profiles.
2. Click the New Profile button.
3. Enter an identifying code for the profile in the Code field.

   Enter a description for the profile in the Description field.
4. Select the type of profile you are creating in the Type drop-down list.
5. Click Finish to continue.
6. To add custom fields to the profile:
   - Select a field code in the Custom Fields drop-down list.

- Select the Required check box if this custom field must be filled in for every record that is created using this profile.

  If a custom field is marked as Required and the user does not enter a value when creating a new record, the system will not allow the user to proceed with the record entry until that field has been filled in. If it is not marked as Required then the user may skip the field.

- Click the Add button.
- Re-order the fields on the profile by clicking the up or down arrows to the right of the custom field.

7 Repeat step 6 for every custom field to be added, the use the navigation links at the top of the page to return to the search screen.

9 Types

Every transaction in Siebel Incentive Compensation Management is assigned to a transaction type. The transaction type dictates which transaction header profile and transaction line profile will be used to generate custom fields for the transaction's header and detail lines. Transaction line types can also dictate which line profile to use for these detail lines (see below).

Transaction types are also used as eligibility criteria by credit rule sets. Every credit rule set is assigned a header profile and a line profile. Since transaction types are also assigned a header profile and a line profile, the credit rule set will check for matching profiles. Transaction line types can also be associated with a line profile (see below). Any transaction type that uses both of the credit rule set profiles becomes a valid transaction type for that rule set. Thus any transaction associated with that type may be passed into the rule set and into the set's individual rules. Transactions types with line profiles that differ from the credit rule set profile are checked for a valid profile associated with the transaction line type. If neither line profile matches the line profile associated with the credit rule set, the transaction is blocked from that rule set and the transaction lines are not credited according to any rules in that rule set. It is recommended that Transaction Types be created in the first working period of your fiscal year, as they are versioned entities.

Setting Up Transaction Line Types

You set up transaction types in the Transaction Types view.

To set up a transaction type

1. Click Configure on the Main Navigation bar, then click Transaction Line Types
2. Click the New Transaction Type button.
3. Enter an identifying code for the transaction type in the Code field.
   Use the Description field to give the type a descriptive name.
4. Select the associated Header Profile and Line Profile.
5. Click the Finish button to complete the transaction type.

Transaction Line Types

Transaction lines may be assigned a transaction line type. The line type can determine the transaction line profile for that line if you need a different set of fields mapped to the credit for that type of transaction line. The transaction line type is primarily referenced by credit rules and can be used to determine credit rule eligibility for transaction lines. It is recommended that Transaction Line Types be created in the first working period of your fiscal year, as they are versioned entities.

NOTE: A unique credit rule set may be required when using a profile associated with the transaction line type and a profile associated with a transaction type.

Setting Up A Transaction Line Type

You set up and work with Transaction Types in the Transaction Line Type view.

To set up a new transaction line type

1. Click Configure on the Main Navigation bar, then click Transaction Line Types.
2. Click the New Transaction Line Type button.
3. Enter an identifying code for the transaction line type in the Code field.

   Use the Description field to give the line type a descriptive name.
4. Select a line profile for this line type (optional).
5. Click the Finish button to complete the transaction type.

Transaction Event Types

Transaction lines are associated with one or more events. An event simply defines when something significant happens in a transaction, such as when a product is ordered or shipped, or when a contract is signed, and so on. Event types define the kinds of events that are significant for your company's transactions. When transactions are created or imported into the system, at least one event is assigned to the transaction and a date is associated with that event. Any number of events may be assigned to new or existing transactions, so long as all events are of different types.

Events also determine how much credit is to be generated for the transaction line. After determining which events on a line are open, the system will look up the transaction's type and use that to determine whether the transaction qualifies for the rule set. If it does, it looks up the event type list that has been set up for that transaction type for that rule set, and looks for the open event on that list. Each event type will have a percentage value associated with it. When the system generates credits for that event, credit values will be multiplied by that value to determine the actual amount of credit that participants or organizations receive. This allows participants to earn partial credit on transactions, ultimately earning 100% credit when all events have been entered and processed. Or, participants can earn full credit on a transaction for every qualifying event, allowing for multiple credits to be generated from a single transaction line. It is recommended that Transaction Event Types be created in the first working period of your fiscal year, as they are versioned entities.

Setting up Event Types

You set up Event Types in the Transaction Event Types view.

To set up event types

1. Click Configure on the Main Navigation bar, then click Transaction Event Types.
2. Click the New Transaction Event Type button.
3. Enter an identifying code for the event type in the Code field.
   Enter a description of the event in the Description field.
4. If this transaction event type will be used for credit rule set qualification, keep the Qualifies for Crediting checkbox checked.
5. Click the Finish button to complete the transaction event type.

Setting up Incentive Types

Incentive types are the types of earnings your participants receive and can be used in formulas to identify different portions of payout being calculated.

To set up incentive types

1. Click Configure on the Main Navigation bar, then click Incentive Types.
2. Click the New Incentive Type button.
3. In the Code field, enter a unique identifying code for the incentive type.
   Use the Name field to enter the full name of the incentive type.
4. Check the Monetary Flag checkbox if this incentive type will be paid to the participant in money. If this box is unchecked, then the system will assume that they payout will be in some other form other than money. To review payouts that are non-monetary, select Payout on the main menu and select the participant type in the Incentive Payouts section of the menu.

   NOTE: The Plan Summarization service will only summarize payouts that are associated with an incentive type that has the Monetary Flag checked.
5. Click the Finish button to complete the incentive type.

10 Hierarchy Levels

Level sets organize the different levels of organizations, territories, and products in your system. When new organizations or territories or products are defined in the system, they are assigned to a specific level. The assigned level determines where in a hierarchy the entity may reside; lower level order numbers indicate higher positions in the hierarchy, and higher-level numbers indicate lower levels in the hierarchy. Organizations with high-level numbers must report to organizations with lower order numbers, and products with high-level numbers can be sub-products or sub-categories of products with lower order numbers. Note that when territories are created, the system will automatically assign the object to the lowest possible level as Territories cannot have children, so it is not necessary to create a level in the territory hierarchy level set, only in the regions that will be parents of territories.

Hierarchy levels must be defined before organizations, products, regions, or hierarchies can be defined, as the level is a required field for organizations, regions, and products, and it is impossible to construct a hierarchy without levels.

NOTE: You need to create a level set for each type of hierarchy, for example, one for organizations, regions, and so on.

Creating Hierarchy Levels

You create Hierarchy Level Sets in the view of the same name.

To create hierarchy level sets

1. Click Hierarchies on the Main Navigation bar, then click Hierarchy Level Sets.
2. Click the View icon next to the level set you wish to modify.

   You may choose Organization Level Set, Territory Level Set, or Product Level Set.
3. Click the Edit button to add levels to this level set.
4. In the Order field enter a number to indicate the level's order.

   Remember that lower numbers will indicate higher levels, while high numbers indicate lower levels. All level numbers must be larger than zero.
5. Enter a code for this level in the Level Code field.
   Note that level codes must be unique within a level set, but you may re-use level codes in different level sets if desired.
6. Enter a Level Description.
7. Click the Add button to add the level.
8. Repeat steps 4 through 7 for each level you need to define for this set. When you have finished adding levels, click Save.

11 Extended Attributes

Extended attributes are data fields that can be added to many Siebel entities and their corresponding screens. They extend the data that can be recorded for certain types of records and allow your company to keep complete track of all relevant data.

Extended attributes are typically referenced by rules to determine plan eligibility or credit rule eligibility conditions. For example, if an extended attribute were created for employees, that attribute could be used to determine which plan the employee qualified for. An extended attribute added for products might be referenced by credit rules to determine whether a transaction was eligible for that rule or not.

Extended attributes may be created to extend the data fields for the following types of records:

- Customers
- Channel Partners
- Channel Segments
- Employees
- Employee Jobs
- Jobs
- Organizations
- Products
- Rate Groups
- Salary Grades
- Cost Centers
- Locations
- Payroll Systems
- Territory

Extended attributes should not be confused with custom fields, which are used exclusively by profiles to generate data fields for transactions, credits, and goals. Also, extended attributes are defined exclusively for the data tables and user interface screens that they will modify; custom fields can be referenced by any profiles. Extended attributes should also not be confused with context attributes, which are used by formulas to determine what types of data may be used in formula components.

Data Types

Extended attributes may be strings (text values), numbers, or dates. For any data type you may define a default value. This default value is automatically entered for the extended attribute field when creating or editing records, although it may be changed or edited at any time.

For any data type you may specify that the extended attribute field is a required field. If a user does not enter a valid value in the field during record creation, the user will not be able to proceed until that field is filled in.

Display Types

For any of the above data types, each extended attribute also has a display type. The display type governs how the data field will appear in the user interface, and may also act to constrain entries to a particular list of valid values or to a range of acceptable values. The display types are as follows:

- **None**. This simply displays the field as a text field or text area field. There are no constraints on what values may be entered in the field, other than the value must match the field's data type.
- **Defined List of Acceptable Values**. This option turns the field into a drop-down list, from which users must choose one of the list values. If you choose this option, you will have to define the list of valid values before finishing the extended attribute.
- **Range of Acceptable Values.** This option displays the field as a text field, and entries must fall between (or be equal to) the endpoints of a specified range. The range must be defined when you choose this option. This option can only be used with numeric fields, not string or date fields.
- **Regular Expression**. Regular expressions are used to constrain entries to particular patterns of text or numbers. Using regular expressions is similar to using wildcards in a search field. The expression consists of regular alphanumeric characters mixed with special characters that stand in for any alphanumeric character or special expressions that validate the type of text being entered. The expression must be defined at the time this option is chosen.

For all options other than Defined List of Acceptable Values, you may set the field to display as a text field or as a text area. A text field is a single line field in which text, numbers, or dates may be entered. A text area is a large text box in which multiple lines of text may be entered. Text areas are commonly used for comment or description fields in which the user is allowed to enter long strings of text. They are also useful for script fields.

Original System Attributes

The Extended Attributes interface can also be used to modify the attributes of original system fields in Siebel Incentive Compensation Management. This can be useful if you want to constrain certain fields to a particular list of values or to a specific range of potential values. For example, the City field on the Employee setup screen is an open text field, in which any city name can be entered. You could convert this field into a drop-down list of specific city names, which may assist users in setting up new employee records.

To change the attributes of an original system field, you must know its Object Query Language (OQL) name. This name is how Siebel Incentive Compensation Management refers to the field throughout the system and does not necessarily match the label displayed on-screen for that field. The OQL name can be found by examining the mapping XML text file that corresponds to the function or screen you are modifying. These XML files have a name of the format *Mapping.xml, where * represents the name of the function or screen, such as *employee* or *formulaComponent*. They can be located in the mapping directory of your Siebel Incentive Compensation Management system directory. Once you have opened the file, finding the correct OQL name for the attribute you want to modify is a fairly easy matter. Most object names are very similar to the field labels displayed on-screen, and the object name is always preceded by sql name=. For example, consider the XML fragment below, from the employeeMapping.xml text file:

```
<field name="homeAddress1" type="string">
<sql name="home_address1" type="varchar"/>
<bind-xml name="homeAddress1" type="string"  node="element"/>
</field>
```

This set of statements refers to the homeAddress1 field for the Employee screen, which is labeled on-screen as *Address 1*. The OQL name can be obtained from the sql name="home_address1" statement. This name, *home_address1*, is how you would refer to the field when modifying its attributes.

Creating Extended Attributes

You create extended attributes from the Extended Attributes & Constraints view.

To create extended attributes

1. Click Configure on the Main Navigation bar, then click Extended Attributes & Constraints.
2. Choose the entity (data type) you want to extend.
3. In the New Name/Label field, enter the name that you want to assign to this extended attribute.

   This is the label that will be attached to the extended attribute data field in the user interface.
4. Verify that the New Extended Attribute check box is selected, then click Next to continue.
5. Choose the data type for this extended attribute field.

   You may choose String, Number, or Date.
6. In the Default Value field, enter the default value that will be automatically filled in the field if the user does not enter or choose another value when setting up a new data record.

   Although a default value is not required, it can often help a user enter a valid value.
7. In the Fill Existing Entities with this Value field, enter a value that will be automatically filled in for any records that already exist for this entity type. This does not have to be the same value as the Default Value. This value only applies to existing data records and will not apply to new records.

8 If the user must enter or choose a value for this field before a data record can be saved, select the Attribute is Required check box.

9 Click Next to continue.

10 Choose the value constraint for the extended attribute field:

- None
- Defined List of Acceptable Values
- Range of Acceptable Values – In the adjoining fields, enter the lowest and highest endpoints of the accepted range of values. This option can only be used for numeric extended attribute fields.
- Value Must Match Regular Expression – In the adjoining field, enter the expression.

11 Click Next to continue.

12 If you chose Defined List of Acceptable Values in step 10:

- Click Next in the next screen.
- Select Create New List to create a new value list and click Next.
- In the Value field, enter the first valid value that will appear in the list. Click the Add To List button to add the value to the list. Repeat this step for every value that needs to appear in the list.
- Click Next to continue, and skip to step 14.

13 If you chose any display type other than Defined List of Acceptable Values in step 10, choose either Text Field or Text Area for the display type, then click Next to continue.

14 If the system verifies that the attribute was successfully added, the extended attribute is finished. If the system indicates failure, then you must re-create the attribute and correct any potential errors that may have caused the failure.

Editing Extended Attributes

You edit extended attributes in the Extended Attributes & Constraints view.

To edit extended attributes

1 Click Configure on the Main Navigation bar, then click Extended Attributes & Constraints.

2 Choose the object (data type) for which you want to modify an existing extended attribute.

3 Select the extended attribute from the displayed list of attributes.

4 You may change the extended attribute's data type from one type to another. Note that making a change in an attribute's data type may invalidate any entries that have already been made in this field. For example, if you switch the data type from String to Number, any entries that contain letters or other non-numeric characters will no longer be valid for this attribute field. In general, however, you can always safely switch from Number or Date to String without invalidating existing record entries.

5 You may change the Default Value for the extended attribute. Note that the new default value will only affect new record entries and will not affect any records for which another value or no value was entered.

6 In the Fill existing objects with this value field, you may enter a new value that will replace all existing entries for this attribute field.
If you have changed the data type for the extended attribute, this field is especially useful for converting invalid entries to valid values. Note that this will apply to all records that use this extended attribute field, and changes to individual records must be made through the appropriate edit screens.

7 Click Next to continue.

8 You may change the attribute's display type to any other type. Note that doing so may invalidate existing entries for some records. For example, if the field previously had no specific display type and you assign it the Defined List of Acceptable Values type, any records with entries that do not appear on the list you create will become invalid entries.

9 Click Next to continue. If you changed the display type you may need to either add a value list (see above) or choose to make the field display a text field or text area. Use the navigation links at the top of the page to return to the search screen.

Editing Original System Attributes

You edit the system attributes that came with your application in the Extended Attributes & Constraints view.

To edit original system attributes

1 Click Configure on the Main Navigation bar, then click Extended Attributes & Constraints.

2 Choose the object (data type) for which you want to modify an original system attribute.

3 Enter the Object Query Language Name of the original attribute.
The OQL name is obtained through an XML mapping file, as described above. Note that OQL names are case-sensitive.

4 You may change the original attribute's data type from one type to another. Note that making a change in an attribute's data type may invalidate any entries that have already been made in this field. For example, if you switch the data type from *String* to *Number*, any entries that contain letters or other non-numeric characters will no longer be valid for this attribute field. In general, however, you can always safely switch from *Number* or *Date* to *String* without invalidating existing record entries.

5 You may change the Default Value for the attribute. Note that the new default value will only affect new record entries and will not affect any records for which another value or no value was entered.

6 In the Fill existing objects with this value field, you may enter a new value that will replace all existing entries for this attribute field.
If you have changed the data type for the original attribute, this field is especially useful for converting invalid entries to valid values. Note that this change will apply to all records, and changes to individual records must be made through the appropriate edit screens.

7 Click Next to continue.

8 You may change the attribute's display type to any other type. Note that doing so may invalidate existing entries for some records. For example, if the field previously had no specific display type and you assign it the Defined List of Acceptable Values type, any records with entries that do not appear on the list you create will become invalid entries.

9 Click Next to continue. If you changed the display type you may need to either add a value list (see above) or choose to make the field display a text field or text area. Use the navigation links at the top of the page to return to the search screen.

12 Labels

Labels are used throughout Siebel Incentive Compensation Management to identify data fields, screen names, screen sections, links or buttons, and error messages. All such objects have standard labels or text defined for them; these are simply text strings that are rendered in the appropriate style for display on the screen. Most labels can be modified to accommodate your company's business practices or terminology.

Page specific labels apply to specific screens for a particular object. Such pages include the *View* page, *Search* page, the *Edit* page, and so on. Following the prior example, suppose you want to remind your users, when adding new locations, that the Location ID number should contain exactly seven digits. For the *Add/Edit* page specific label for Locations, you would change *Location ID#* to read something similar to *Location ID# (must be 7 digits)*.

Editing Page Specific Labels

You edit page specific labels in the labels screen.

To edit page specific labels

1. Click Configure on the Main Navigation bar, then click Labels.
2. In the Select a Section to Edit list hover over the choices, then choose a screen type. For instance Company > Calendar.

   The Edit Label Group view appears.
3. Choose the specific type of page for which you want to edit labels.

   These pages apply only to the object you have chosen, and not to similar pages throughout the system.
4. Locate the label you want to change in the Current Value column. You may perform a search on the label name, or you may perform a Find and Replace operation by entering text into the Find and Replace text boxes.

   NOTE: The label names in this column will change if you make modifications. For any page specific label that has been changed, you may click the Revert button for that label to revert back to the original label name.
5. Click the field to make it active, then edit the text directly into the field.
6. Repeat the steps each label you want to change.
7. Click Save to retain your changes.

13 Context Attributes

Context attributes provide the means to reference any data entity in the system within credit rules, plan rules, and calculation formulas. They significantly extend the power of such functions and are an essential key to building complex and highly customized incentive plans within Siebel Incentive Compensation Management.

Context is an important concept in crediting and calculation functions. It defines the scope of such functions and defines what is automatically known to the system during each of these processes. During crediting, for example, credits are created within the context of one transaction line event and within the context of the current calendar period. Thus, when the system processes a transaction line event through a credit rule, it can automatically refer to any information found on the line event or the transaction header, and automatically knows what the current period is (in case the rule's conditions or distributions are period dependent). Naturally, this means that any other data outside of this scope is not within the current context and thus ordinarily cannot be referenced by that process. The product sold on a transaction line is within the context of the transaction line event, but any other details about that product are not within that context.

To make credit rules and formulas more flexible and powerful, additional *context variables* can be defined to essentially bring additional data within the context of the credit rule or formula. A context variable is a custom script that refers to a specific data table and a specific field on that data table, and retrieves the correct data according to the current context.

For example, without using context attributes, a credit rule will only be able to reference data within a specific transaction record. A rule could refer to the product sold on a transaction line as part of the rule's conditions, for example, but it could not refer to any other data about that product because that information is not on the transaction record. To solve this problem, you would set up a context attribute for the crediting service that specifically refers to the Product table and a specific data field on that table, such as the product's unit price or default rate. Then, when setting up the credit rule, you could set up conditions using those context attributes and select transaction records based on the product sold and that product's price or rate. You could also refer to these context attributes when creating distributions so that attributes of product records are copied to a participant's credit record. The system will know which product's attributes to refer to because the specific product is already defined within the context of the transaction line.

As another example, without using context attributes, calculation formulas can only refer to performance record data for a specific employee or the results of other formula components. The formula could reference credit and goal records for an employee, but could not look up any basic information about that employee, such as his or her job function or the rate group he or she belongs to. Again, you would set up context attributes to get around this problem. In this case, you would create a context attribute that refers to one of the Employee data tables and specifically to one of the data fields on those tables, such as the employee's assigned job code. Within the calculation formula, you could then refer to the *job function* context attribute within any component and use that job code in whatever manner required. Again, the system will know which specific employee's record to refer to because the employee is already defined within the context of the plan for which the formula is being calculated.

Context attributes can be assigned to the system and made available to all OUs in the system or they can be assigned to a specific OU and used solely for that OU processing needs. If a context attribute is assigned to a specific OU, it will become part of the OU export (See "Operating Unit Export and Import"). Once it is included in the files for the OU export, it can be imported into a modeling scenario or into another instance of Siebel Incentive Compensation Management.

The meat of a context attribute is a JavaScript script. This script tells the system which data table to get data from, how to look up the correct record, and any other calculations that need to be performed to get the required data. For example, a simple context attribute could look up the date on which an employee started a particular job function. A more complex script could look up the employee's job start date, look up the current date from a system calendar, and then calculate the difference between the two dates to figure out how many days the employee has held his or her current job function.

Only attempt to set up a context attribute if you are familiar with using JavaScript. The context attributes that your company needs to execute its incentive plans should have already been set up during the initial implementation of Siebel Incentive Compensation Management. If you find that you need to create new attributes and require assistance, contact your Siebel Technical Support representative or refer to Appendix A for examples of existing context attributes.

Defining Context Attributes in Incentive Compensation Management

To define context attributes

1. Click Configure on the Main Navigation bar, then click Context Attributes.
2. Click the New Context Attribute button.
3. Enter a name and description for the attribute in the Name field and the Description field respectively. This name will be referenced by rules or formulas when you choose to include the context attribute within a rule or formula.

   NOTE: Because context attributes are assigned to specific services, it is possible to reuse the same name for two attributes if they are associated with different services. However, you cannot create two attributes with identical names that are assigned to the same service.
4. In the Service Code field, select the service to which this context attribute will be assigned. Context attributes cannot be shared across services, although you may create context attributes that perform the same function but are assigned to different services.
5. Select a Data Type for the context attribute.

   The data type tells Siebel Incentive Compensation Management what kind of data it should expect to return when it runs the script. Options are:

   - **Boolean**. The result should be True or False.
   - **String**. The result may be any text string of alphanumeric characters.

- **Number**. The result should be a number. Note that a common mistake in creating JavaScript scripts is to generate a result that looks like a number but that is actually formatted as a text string.
- **Currency**. The result should be a number in currency format.
- **Date**. The result should be a date in the MM/DD/YYYY format.

6 Select the Cumulate checkbox if this context attribute will be cumulating the data contained in the variable.

7 Uncheck the System Owned checkbox if this context attribute will be specific for this operating unit.

NOTE: System-owned Context Attributes are not included in Operating Unit exports. Non-System-owned Context Attributes are included in Operating Unit exports and are created in the target generating unit when the export set is imported into another Operating Unit.

8 Choose either an Existing Category to assign to the attribute, or enter a New Category for this attribute.

Categories help to organize context attributes according to their services, the data they reference, their functionality, and so on. If you set up a new category for this attribute, that category will be available as an existing category for all future context attributes.

9 Enter your script in the Script text area.

You may also copy and paste an existing script into this field.

NOTE: Refer to Appendix A for examples of context attribute scripts.

10 Click the Finish button to complete the context attribute.

14 Templates

Condition templates are used to build the selection conditions found in credit rules and plan calculation eligibility rules. The template specifies what data is to be examined, what to compare it to, and what condition or conditions the data must meet in order to be eligible for that rule.

Several basic condition templates are provided with Siebel Incentive Compensation Management for use in credit rules and plan rules. These templates cover many of the most common conditions that will be used in rules, and can be used in credit rules or plan rules without modification.

Customized condition templates can be set up to handle more complex condition test structures. For example, a condition template could be set up to test two or more separate conditions, and return a value of *True* if any one of the conditions is satisfied. This can be used to effectively create OR conditions within a credit rule or plan rule. Normally, credit rules and plan rules consider all conditions to be joined by AND functions, meaning that all conditions must return a value of *True*. This type of customized condition template gets around this limitation by incorporating all OR conditions into a single template.

Once a template has been set up for either plan rules or credit rules, that template is available for use in all plans or credit rules. Templates can be assigned to the system and made available to all OU's in the system or they can be assigned to a specific OU and used solely for that OU processing needs. If a template is assigned to a specific OU, it will become part of the OU export (See "Operating Unit Export and Import"). Once it is included in the files for the OU export, it can be imported into a modeling scenario or into another instance of Siebel Incentive Compensation Management.

Setting Up Condition Templates

You set up condition templates from the view of the same name.

To set up condition templates

1. Click Configure on the Main Navigation bar, then select Condition Templates.
2. Click the New Condition Template button.
3. Enter a unique ID for the template in the Code field.

   This code should describe the template's purpose so that users can easily choose the correct template within credit rules or plan rules.
4. Enter a name for the template in the Name field.

   This name should fully describe what the template accomplishes.
5. In the Service Code field, select one of the following:

   - **Plan Calculation Service**. This associates the template with plan rules, which use conditions to test participants for eligibility.

- **Sales Crediting Service**. This associates the template with credit rules, which use conditions to test transactions and transaction lines for eligibility.

6 Uncheck the System Owned checkbox if this template will be specific for this operating unit.

NOTE: System-owned Context Attributes are not included in Operating Unit exports. Non-System-owned Context Attributes are included in Operating Unit exports and are created in the target generating unit when the export set is imported into another Operating Unit.

7 Click Next to continue.

8 Click the Descriptive Text button, then enter all or part of a description of the template in the Enter Text Below field and click the Add button.

This description is what users will see when they add a condition template to a plan or credit rule, and should describe exactly what the condition will do.

9 Add other elements to the template as follows:

- **Object Type**. This adds an object input to the template. Objects are standard data fields found on transactions or participant records. They may also be any data fields that are automatically available for a particular service, such as the current calendar period. After choosing this element, select the specific object in the Select Type field, then click the Add button.
- **Text Input**. Inserts a text input field into the template.
- **Numeric Input**. Inserts a numeric input field into the template.
- **List**. Inserts a drop-down list into the template.
- **Operator**. Puts an operator selection list into the template. Operators generally include any of the standard condition operators (less than, equal to, not equal to, etc.).

10 At any time you may click the Remove Last Line button to remove the last element in the template string.

11 Click Next to continue.

12 In the Condition Template:JavaScript field, the system displays the JavaScript code that is automatically generated by your element selections.

In this field, enter additional script lines that define what the system should do with its input values and what conditions will return a value of True and what will return a value of False.

NOTE: Refer to Appendix B for examples of condition template scripts and tips on setting up new scripts.

13 Click Finish to complete the template.

About Recipient Templates

Recipient templates are a way of grouping together a certain set of participants and organizations for the purposes of creating credit rule distributions. The recipient template is a set of conditions that select particular employees, related channel partners or customers, organizations, and so on. The template can then be referenced within a credit rule distribution. When credits are generated according to that rule, the system will determine which participants or organizations fit the conditions of the recipient template, and generate credits for each one. Each such recipient receives the same amount of credit; essentially, one credit record is copied amongst all the listed recipients.

It is possible to define several sets of recipient templates and to call several templates within a single credit rule by creating one distribution for each recipient template.

Several standard recipient templates are provided with every Siebel Incentive Compensation Management implementation, and can be used in any credit rule distribution.

Setting Up Recipient Templates

You set up recipient templates in the Credit Recipient Templates view.

To set up recipient templates

1. Click Configure on the Main Navigation bar, then select Credit Recipient Templates.
2. Click the New Credit Recipient Template button.
3. Enter a unique ID for the template in the Code field.

   This code should describe the template's purpose so that users cant easily choose the correct template within credit rule distributions.
4. Enter a name for the template in the Name field.

   Recipient Templates have a system-owned attribute similar to Context Attributes and Condition Templates.
5. The only available service for this template type is the SalesCreditingService.
6. Click Next to continue.
7. Click the Descriptive Text button, then enter a full or partial description of the template in the Enter Text Below field and click the Add button.

   This description is what users will see when they add a credit recipient template to a distribution, and should describe exactly how recipients will be selected.
8. Add other elements to the template as follows:
   - **Object Type**. This adds an object input to the template. Objects are standard data fields found either on transactions or participant records. After choosing this element, select the specific object in the Select Type field, then click the Add button.
   - **Text Input**. Inserts a text input field into the template.
   - **Numeric Input**. Inserts a numeric input field into the template.

- **List**. Inserts a drop-down list into the template.
- **Operator**. Puts an operator selection list into the template. Operators generally include any of the standard condition operators (less than, equal to, not equal to, etc.).

9 At any time you may click the Remove Last Line button to remove the last element in the template string.

10 Click Next to continue.

11 In the Credit Recipient Template:JavaScript field, the system displays the JavaScript code that is automatically generated by your element selections.

In this field, enter additional script lines that define what the system should do with its input values and how to select recipients according to those values.

NOTE: Refer to Appendix C for examples of condition template scripts and tips on setting up new scripts.

12 Click Finish to complete the template.

15 Services Overview

The Service Manager works as the central console for managing, and reviewing all processing services. Clicking Services on the Main Navigation bar, then clicking All Services will open the Service Manager.

Import Services

Siebel Incentive Compensation Management uses standard import specifications for bringing data from external applications into the Siebel Incentive Compensation Management database. These processes can import most entities in this system including:

- Participants
  - Employees
  - Customers
  - Channel Partners
- Job Codes
- Location Codes
- Cost Centers
- Salary Grades
- Transaction Data
- Product
- Performance data
  - Goals
  - Credits
- Organizations
- Territories
- Currency exchange rates
- Credit Rule sets
- Matrix components
- Step Calculation Components
- Threshold Components
- Formulas
- Plans

When the user chooses an import service to run, the system will ask the user to browse for the path and name of the target import file. The import file's name must always be in the format **.xml,* where * represents the file's name, and *xml* indicates that the import file is an XML document. See "Open Integration Framework" for a list of the allowed XML data files and their structures.

The import process works as follows:

1. The system verifies that the chosen file is in the location specified and that it is an XML document. If the file cannot be found or it is not a valid XML file, the system returns an error and the process stops.
2. The first record within the XML record set is opened, and the system verifies each data line in the record to make sure that all required data fields contain data, and that no field contains invalid data.
3. If the verification passes, the system writes each data line in the record to the corresponding data field on the target database table.
4. If a record contains invalid data, or is missing data from certain fields, the system sends the record to an error log, which also records the reason for the import failure. The record is not written to the database at this point.
5. Steps 2 through 4 are repeated for each record in the XML record set.
6. If at any time the number of errors exceeds the limit specified for this particular service (see *Error Handling* below) the system will stop the process and notify the user of the problem.

Service Logging

Each service controlled by the Services Manager in the service launcher has six different settings to log processes and errors when the services are running. These log levels include Default, Debug, Info, Warn, Error, and Fatal. Each time the service manager is accessed, the log level for the services will be reset to default. The default log level is defined by the service.logging.default property in the files <INSTALL_DIR>/Config/SiebelICMConfig.xml.in and can be set to one of the logging levels: Debug, Info, Warn, Error, or Fatal.

The setting selected in the services manager will not alter any of the log file settings described in the previous section.

- Default – This is the default setting for all services. Provides error messages when errors occur and processing times when no errors are present.
- Fatal – Provides severe error messages and processing times.
- Error – Provides all error messages and processing times. Use this setting when reporting issues to Siebel Technical Support.
- Warn – Provides all error messages, warning messages, and processing times.
- Info – Provides all error messages, warning messages, information messages and processing times.
- Debug – Provides all the information from all other settings as well as step-by-step processing information. Use this setting only at the direction of Siebel Technical Support.

Sales Crediting Service

Any incentive plan that makes use of transaction records will run this process to generate and distribute credit records. The process runs as follows:

1. The system opens the first credit rule set, according to the credit rule set codes. Note that it does not matter in what order the system processes rule sets.
2. Each transaction's type is checked against the valid transaction types for the rule set (according to the header and line profiles of each). If the transaction matches one of the valid types it gets passed to the next step, otherwise it will not be passed into the rule set.
3. Each transaction line's events are checked against the event eligibility conditions of the rule set for that transaction type. If the transaction contains any open events that match the event conditions of the rule set, then the transaction line is passed into the rule set. If all events on the transaction are closed, or if there are open events on the transaction line that are not valid according to the event eligibility criteria, then the line is not passed into the rule set.
4. Any transaction lines with events that qualified for the rule set are passed on to the first rule within the rule set, according to the credit rule codes.
5. The first available transaction line is tested against the conditions of the credit rule. If the transaction line fails to match any one condition, it does not qualify for that rule; such transactions are set aside to be passed on to the next rule (step 7).
6. If the transaction line that does qualify for the credit rule, the system generates credit records according to the distributions set up for that rule. Distribution records are generated in this manner for each open event on the transaction line. This transaction line is then set aside and not passed on to any more rules within the rule set. The system then gets the next available transaction line and returns to step 5 for that line.
7. Once all transaction lines have been tested against the rule's conditions and distributions made as appropriate, the system goes to the next credit rule in the set and returns to step 5. Any transaction lines that did not qualify for the previous rule are now tested against the new rule according to steps 5 and 6.
8. Steps 5 through 7 continue until either all transaction lines have been credited or until no more rules are left in the set. Either condition signals the end of the rule set.
9. The system goes on to the next rule set and repeats steps 2 through 8 for that set. This continues until all rule sets have been processed.

The Sales Crediting service can be run in either *Full Batch* mode or *Incremental* mode. In Incremental mode, the system only processes transaction lines and events for which no credits have yet been generated. Full Batch mode, on the other hand, processes all transaction line events that have not yet been closed out by the Close Period process. This includes all transactions that have already had credits generated for them, even if payout calculations and summarizations have already been calculated in the current period. The Full Batch mode process will only exclude transaction line events that have been credited and paid in prior processing periods, as those events should be marked as "closed" by the Close Period process.

For most instances of the Sales Crediting service the user will only want to process transactions in Incremental mode, as this will speed up the processing time considerably. This will allow the user to only credit newly entered transactions, transaction adjustments, and so on. Credits should be processed in Full Batch mode when significant changes to credit rules or rule sets are made in the middle of a processing period and those changes must affect any transactions that were already processed according to the older rules.

Rollup Service

If any measures have been set up to roll credits up through the organization hierarchy, this process handles that function. The process runs as follows:

1. The system starts at the bottom level of the organization/territory hierarchy, finds organizations/territories for which credits have been generated in the current period, and starts with the first organization/territory (in alphanumeric order).
2. If the selected organization/territory has any credit records for which rollups must be made (according to the credit's measure's specifications) then the process continues as outlined below. If no such records exist the system skips to the next organization/territory.
3. For the first qualifying credit record, the system examines the associated measure's rollup specifications to find the top-most level of the rollup and to find the organization/region directly above the current organization/territory. If the current organization/region level happens to be the top-most level of the rollup, then the system does not process rollups any further for this credit record and moves on to the next available record.
4. The system copies the credit record to the organization/region directly above the current organization/territory, according to the hierarchy.
5. Steps 3 and 4 repeat for every record that must be rolled up. In each case the associated measure may have a different rollup specification, and the system will have to check the structure in each case to find the top-most level of the rollup.
6. Once all required credits have been rolled up for the organization/territory, the system moves on to the next organization/territory at this level of the hierarchy, and repeats steps 2 through 5.
7. After all organizations/territory at this level have had credits rolled up, the system goes on to the next level in the hierarchy and repeats steps 2 through 6.
8. The process ends when the system has processed all necessary rollups for the period.

Note that as the process continues, any credit records that were rolled up to the current organization/region from a lower level, will be rolled up to the organization/region at the next higher level, assuming that the current organization/region isn't the top-most level of the rollup.

This process also rolls up goal records, if any measures are set up to rollup goal data.

Cumulate Service

If any measures have been set up to cumulate credit records, this process handles that function. The process runs as follows:

1. The service examines each credit's measure and only processes the credit if the measure has any cumulation instructions specified.
2. The system finds each of the custom fields on the credit record and processes cumulations for each field separately.
3. For the first custom field, the system finds that field's aggregate operator. If the field has no operator specified, then no cumulations are processed for that field and the system goes on to the next custom field.
4. According to the frequency group and data group of the measure, and the mathematical function specified by the aggregate operator, the system calculates the cumulated value for the current custom field for the current period.
5. The system repeats steps 3 and 4 for each custom field on the credit record.
6. Steps 1 through 5 are repeated for each credit record in the current period.

This process also cumulates goal records if measures have been set up to cumulate goals as well.

The Cumulate Service should always be run after the rollup process, if both rollups and cumulations are part of your incentive plans. This makes sure that rolled up credit records will be cumulated for all organizations/territories or regions. The Rollup Service only rolls up base credits, not cumulated credits.

Plan Calculation

The Plan Calculation services processes your incentive plans. The PlanEligibilityService matches participants to plans, The Plan Calculation service calculates formula results for each participant, and the Plan Summary service summarizes the calculation results into payout. Plans are processed in alphanumeric order. The process flows as follows for each plan:

1. The PlanEligibilityService finds all participants that match the Participant Type chosen for each plan and assigns them to that plan.
2. Each participant is matched against the plan's rules. If the participant does not meet all of the plan's conditions, the system skips that participant and moves on to the next one. If the participant does qualify, the system proceeds with calculation as outlined below when the PlanCalculationService is run.
3. The PlanCalculationService gets the first calculation formula listed for the plan. For this formula, the system loads all necessary data variables, including the participant's performance records, participant records, product records, and so on. This data is then processed through the formula's components to generate a calculation result. Once this result has been recorded, the system repeats this process for each calculation formula listed for the plan.
4. If a draw has been specified for the plan or for the participant, the system determines whether or not the summarized payout result meets the draw's criteria. If it does, the system calculates a draw adjustment and adds this to the payout result. If no draw adjustment is necessary but draw recovery is possible, the system calculates how much it can recover from the employee and subtracts this amount from the summarized payout result.

5 If a cap has been specified for the plan or for the participant, the system determines whether or not the summarized payout result (including any draw adjustments from above) meets the cap's criteria. If it does, the system enters an adjustment to reduce the summarized payout to the cap level. If the cap is set up to carry balances forward, any money lost due to the cap is stored as a positive payout adjustment for the next processing period.

6 Steps 2 through 5 repeat for each participant that matches the plan rules. Once all qualifying participants have been processed through the plan, the system will go to the next plan and repeat this procedure.

7 When the PlanSumaryService is run, if no summary formula has been set up for the plan, the system adds (summarizes) all the results together for the participant. If a summary formula is specified, the calculation results are processed through this formula to generate the summarized result for the participant.

Close Period

The Close Period process finalizes all processes for the current period. This process should only be run if the following is true:

- Transactions, credits, and payout results have been reviewed and approved by the appropriate personnel.
- No further processing is to be done in the current period. This means that all transactions have been entered and accounted for, and no changes need to be made to any credit rules, formulas, or other plan components.

This process accomplishes the following tasks:

- Updates every participant's balance data to reflect payouts and payout adjustments.
- Updates transaction line events and marks as "closed" any event that has been credited and paid upon.
- Deletes any temporary tables that were created during other processes.
- Preserves the current state of the database for future reference. This allows users to switch to a prior working period and view data records from that period.

Update Analytics

This process copies the transactional database to the analytics database. This process can actually be run at any time during a period but is generally useful only at the end of a period, when all transactions have been processed and payouts made.

16 Service Launcher

The Siebel Incentive Compensation Management Service Launcher allows administrators to create a mechanism that will automatically launch services in a particular order and stop these same services based on the number and type of errors encountered. This feature can be used when it is important to import a large amount of data, such as transactions, on a nightly basis. Additionally, this feature can be used to run all the services associated with crediting and plan calc over night, so the compensation administrator can review the results and make any necessary changes during normal working hours.

The service launcher file can be setup for a single period, for multiple periods, or it can be setup to run the same service multiple times. Examples of these setup files can be found in the main level of your Siebel Incentive Compensation Management install directory.

Setting Up a Service Launcher Properties File

To setup a service launcher .properties file simply open an existing file (serviceLauncherExample_Basic.properties, serviceLauncherExample_MultiPeriod.properties, serviceLauncherExample_RepeatingServices.properties edit the values, and save the file anywhere on your hard drive. The properties file contains the following properties:

- **service.runas.username** – This is the operating unit administrator username. If this operating unit administrator has access to more than one operating unit, the services will be run on the operating unit that this used logged into last.
- **service.runas.password** – This is the operating unit administrator password associated with the user name above.
- **service.X.yearCode** – Where 'X' stands for the index (i.e. execution order) of the service, starting with 1 and incrementing by 1. The yearCode must be specified for the first index and is optional for other indexes.
- **service.X.periodNumber** - Where 'X' stands for the index (ie. execution order) of the service, starting with 1 and incrementing by 1. The periodNumber must be specified for the first index and is optional for other indexes.
- **service.X.serviceCode** - One of the following valid values:
  - ChannelPartnerImport
  - CreditImport
  - CumulateService
  - CustomerImport
  - EmployeeImport

- ExchangeRateImport
- GoalImport
- JobImport
- MatrixCalcImport
- OrganizationImport
- PeriodCloseService
- PlanCalculationService
- PlanEligibilityService
- PlanSummaryService
- ProductImport
- PurgePeriodDataService
- RollupService
- RuleSetImport
- SalesCreditingService
- SalesTransactionImport

  NOTE: To use the service launcher for transaction import, the service.extract.newline.numberchars variable in the install.properties file must be set as follows:
  For Unix servers – set this variable to 1
  For Windows servers – set this variable to 2

- TerritoryImport
- UpdateAnalyticsService
- YearCloseService

- **service.X.input** - Represents an optional filename or input text for the service.
- **service.X.haltOnLauncherError** - An optional parameter that tells the ServiceLauncher whether or not to exit if it comes across an exception during the execution of a particular service. By default, it is set to false. Anything other than true is considered false, unless the field is blank. If the field is blank, it haltOnLauncherError remains unchanged from its previous value.
- **service.X.haltOnServiceError** - An optional parameter that tells the ServiceLauncher whether or not to exit if it comes across any failures in a service run execution. By default, it is set to false. Anything other than true is considered false, unless the field is blank. If the field is blank, its haltOnLauncherError remains unchanged from its previous value.

Running the Service Launcher

To start the service launcher

1. Open a command prompt.
2. Navigate to the main level of your Siebel Incentive Compensation Management install directory.
3. Type serviceLauncher.bat *<Path and file name of service launcher properties file>.*

   Example: serviceLauncher.bat <INSTALL_DIR>/serviceLauncherDaily.properties.

17 Open Integration Framework

The purpose of this chapter is to describe the interface to Siebel Incentive Compensation Management importable data entities. It is meant as a guide for implementors and integrators, to help them design import and export files for Siebel Incentive Compensation Management.

The Siebel Incentive Compensation Management Open Integration Framework is designed to accept XML documents, where the XML structure is defined by XML Schemas, one for each type of object listed below.

Basic XML Schematics

The basic pattern of any XML file looks like the following example:

```
<recordset>
    <TYPE>
        <field1>value</field1>
        <field2>value</field2>
        <field3>value</field3>
        <field4>value</field4>
        <field5>value</field5>
    </TYPE>
</recordset>
```

The <recordset> tag indicates the beginning of the XML data, and between this tag and the </recordset> tag are contained all of the files that will be imported.

The <*TYPE*> tag indicates the beginning of a record of a specific type. In an actual file the word *TYPE* would be replaced by a record type such as *EMPLOYEE* or *CUSTOMER*. The paired </*TYPE*> tag indicates the end of a specific record. Between these two tags are any number of <*field*> lines, indicating specific data fields for each record. Following the </*TYPE*> tag would be another <*TYPE*> tag to begin the next record, and this pattern continues throughout the file.

Within each pair of field tags (such as <*field1*> </*field1*>) is a specific value. The term *field* stands for the name of any data field found on a record, such as *code* or *lastName*. The *value* term may be any valid value for that data field; for example, *lastName* would accept any text string for an employee's last name. The number of such fields for any record may vary according to the type of import file, whether or not the data field is required, and so on.

In some types of XML files, there may be "sub-records." Such a file might look like the following:

```
<recordset>
```

```
    <TYPE>
        <field1>value</field1>
        <field2>value</field2>
        <field3>value</field3>
        <field4>value</field4>
        <field5>value</field5>
        <SUBTYPE>
        <fielda>value</fielda>
        <fieldb>value</fieldb>
        </SUBTYPE>
    </TYPE>
</recordset>
```

Here, the <*SUBTYPE*> tag would represent any kind of data within a record that might require its own sub-set of data. For example, every employee is assigned a job function and each job function is associated with a specific set of data. Employees may switch jobs during their career, and Siebel Incentive Compensation Management tracks these as separate job histories. Thus, each job history record for an employee would be stored between separate pairs of <*JOB*> and </*JOB*> tags within the larger <*EMPLOYEE*> record.

Custom Fields & Attributes in XML Files

In addition to the standard field tags in a data file, XML records may contain references to custom fields and custom attributes. The tag for any custom field follows the pattern:

<customField readingType="*Name*" *typeof*value="*Value1*"/>

The readingType tag indicates the name of the custom field, and within the quotes you must specify the actual name of the custom field. The next attribute indicates the type of value contained in the reading type; the tag *typeof*value would actually be replaced by the type of data stored in the custom field, such as stringvalue or numericvalue. The actual value is specified within the quotes.

The tag for a custom attribute follows the pattern:

<extAttribute name="*Name*" value="*Value1*"/>

Similar to custom fields, the name tag indicates the name of the custom attribute, and the value tag indicates the actual value stored in that field.

Note that in both cases, there is no closing tag for either tag. The / at the end of each custom field or custom attribute line indicates the end of that tag automatically.

Any number of custom fields or custom attributes may be contained within a record. They can be added manually to an XML file, of course. In most cases, the export function of external systems should be configured to send the appropriate data into custom fields or custom attributes, or an external XML adapter should be used to transform the exported file into the proper format for import into the Siebel Incentive Compensation Management system.

Thus, for example, an employee record in XML format might look like the following:

```
<recordset>
    <Employee>
        <code>00058</code>
        <lastName>Schmidt</lastName>
        <firstName>Oliver</firstName>
        <officePhone>925-620-2917</officePhone>
        <emailAddress>oliver_schmidt@bigcompany.com</emailAddress>
        <employeeModifier>0.5</employeeModifier>
        <participantFlag>true</participantFlag>
        <hireDate>1996-08-13</hireDate>
        <extAttribute name="birthday" value="1958-07-08"/>
        <Job>
            <code>AE</code>
            <startDate>2001-09-05</startDate>
            <endDate>2001-10-05</endDate>
            <hourlySalary>120.00</hourlySalary>
            <jobStatus>ACTIVE</jobStatus>
            <costCenterID>TIC</costCenterID>
            <orgID>SFBA</orgID>
        </Job>
    </Employee>
… repeat <Employee> structure as many times as needed …
</recordset>
```

Dependency Sorted Import Services

Several import services, such as Organization, Product, Employee, and Territory, have the ability to sort the imported XML file based on parent-child relationships. This will result in the xml being imported in the correct "top-down" order (parents are imported and created before their children). No change is necessary to the order or format of the organization/product/territory xml files - the sort takes place during the import process.

Date Formats

All date (and date-time) values are in the following format:

CCYY-MM-DD

Where CCYY is the 4-digit year, MM is the month number (with leading zero as needed) and DD is the day number (with leading zero as needed).

Example:

1999-01-01 equals January 1, 1999)

2001-03-05 equals March 5, 2001)

2001-05-03 equals May 3, 2001)

Dates that are not in this format in the import file will cause the record to be rejected.

This format is required by the underlying data type "xsd:date" defined in the XML Schema. (See http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#date for the official description of this format.)

Date + Time Formats

If a field needs to support time information as well, the XML Schema should be designed to reflect that fact; the XML Schema design will support this data type wherever necessary.

Ideally the import facility should be much more flexible and support dates in a variety of formats. The internationalization aspect of this makes it rather difficult, and so an internationally accepted date format standard has been chosen.

Employee/Job History/Job

The Employee and Job objects implement Extended Attributes.

Custom Fields and Profiles do not apply to Employee, Employee Job, or Job entities.

Core entities:

- **Employee:** the employee record
- **EmployeeJob:** the association between an Employee and a Job.

At this time, only one job can be included in the import record. All fields in the schema that end in ID represent natural keys (or the 'code' field) of the related object. If you include values for any of these fields, the related record must be present in the database or the record will be rejected. The following is the list of dependent objects that must be present in the database before you can import employee data referencing them:

- Job
- Organization
- Cost Center
- Territory
- Payroll System
- Currency
- Channel Segment
- Rate Group
- Salary Grade
- Location
- Supervisor(Employee) - if you are including SupervisorIDs, which are Employee codes, the supervisor you are referencing must already be present in the Employee table. Obviously there is a dependency here, it may be necessary to import in two passes.
- Role (only if importing user account information)

User Accounts can be created at the same time as employee records by including the desired username, password, and at least one role in the employee XML. The user's role must have been created previously. Passwords are imported in the clear. They will be hashed on import and stored in a hashed form.

Employee Schema (employee.xsd)

The following elements (or fields) are defined in the XML Schema:

NOTE: Required fields in listed in bold text. Cardinality, or number of occurrences of the field in braces: {one or more}. No braces means that the field can occur only once. Attributes of an element are included in <>.

recordset: the root element of the input file.

- Employee
  - code - this is the natural key of the Employee object
  - lastName
  - firstName
  - middleName
  - SSN
  - officeMailstop

- officePhone
- officeFax
- emailAddress
- homeAddress1
- homeAddress2
- homeCity
- homeState
- homePostalCode
- homeCountry
- homePhone
- timeZone
- employeeModifier (string)
- participantFlag - set to true if this employee is a participant
- payEligibilityFlag - set to true if this employee is eligible for pay
- payrollSystemID - (string - validate against PayrollSystem table)
- currencyID - (validate against Currency table)
- channelSegmentID - (validate against ChannelSegment table)
- locationID - (validate against Location table)
- rateGroupID - (validate against RateGroup table)
- hireDate - (string - format CCYY-MM-DD)
- terminationDate - (string - format CCYY-MM-DD)
- employeeStatus
- <extAttribute name - "attribute name" value="my value"/>{zero or more} (must be a declared extended attribute on the EMPLOYEE object)
- Job {zero or one}
  - **Code** - (if a Job element is included, the code must be provided. The code is validated against the Job table.)
  - StartDate - (string - format CCYY-MM-DD)
  - endDate - (string - format CCYY-MM-DD)
  - exemptFlag
  - fullTimeFlag
  - salaryPayPeriod (string)
  - annualSalary (float)
  - hourlySalary (float)

- paidSalary (float)
- jobStatus
- salaryGradeID - (code - validate against Salary Grade table)
- costCenterID - (code - validate against CostCenter table)
- orgID - (code - validate against Organization table)
- supervisorID - (code - validate against Employee table)
- territoryID - (code - validate against Territory table)
- <extAttribute name="attribute name" value="my value"> {zero or more} ("attribute name" must be a declared extended attribute on the EmployeeJob object)
- account - <username="username" password="password">
- roleID - {zero or more} (code of a Role to assign to this user)

Sample XML document for employee.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<recordset>
   <Employee>
      <code>00058</code>
      <lastName>Koka</lastName>
      <firstName>Savi</firstName>
      <officePhone>925-600-2907</officePhone>
      <emailAddress>kjirsten_koka@againtech.com</emailAddress>
      <employeeModifier>0.5</employeeModifier>
      <participantFlag>true</participantFlag>
      <hireDate>1996-08-13</hireDate>
      <extAttribute name="foo" value="13"/>
      <Job>
         <code>AE</code>
         <startDate>2001-09-05</startDate>
         <endDate>2001-10-05</endDate>
         <hourlySalary>120.00</hourlySalary>
         <jobStatus>ACTIVE</jobStatus>
         <costCenterID>TIC</costCenterID>
```

```
            <orgID>SFBA</orgID>
        </Job>
        <account username="skoka" password="mypassword">
            <roleID>Participant</roleID>
        <account>
    </Employee>
… repeat <Employee> structure as many times as needed …
</recordset>
```

Sales Transactions (Also Known as Invoice)

Extended Attributes do not apply to the Sales Transaction core entities.

Profiles: both the Sales transaction Item Type and the Sales Transaction Line Type may have profiles defined. The Custom Fields included in the imported Sales transaction records will be checked against the profile.

Core entities:

- Sales Transaction Item - the master or header record.
- Sales Transaction Line - the line item. Item contains one or more lines.
- Sales Transaction Event - the event associated with the line. A line contains one or more events.

Data Model Comments

In Incentive Compensation Management, the Sales Transaction Item, Line, and Event objects replace INVOICM and INVOIC. Line Types and Event Types are not defined by the application. You are free to define Line Types and Event Types that make sense to the client. For example, if the date on which a contract is signed is significant for crediting purposes, an event type called "Contract Signed" could be defined.

Any number of sales reps may be associated at both the master (or item) level and at the individual line level.

Sales Transaction Adjustments

A sales transaction may be adjusted at the header (item) level, line level, and event level. Adjustment actions are as follows:

- Item level actions = create, cancel, adjust
- Line level actions = add, cancel, adjust, return

- Event level actions = add, cancel

You can indicate the adjustment action to take for the item, line, or event by specifying a value for an attribute called "adjustType". This attribute is present in the SalesTransItem, SalesTransLine, and SalesTransEvent elements (see XML Schema description below). The adjustType values are checked and handled first at the item level, then for each line in the item and each event in the line. The tricky part is telling the import server that the adjustment is only at the line or event level.

The rules for specifying adjustType are as follows:

At the item level:

If adjustType is not present, assume this is a new sales transaction and attempt to create it, along with all lines, events, readings, and sales reps. Do not check adjustType at the line or event level. Same as specifying "create_item".

- if adjustType = "create_item", create a new sales transaction along with all lines, events, readings, and sales reps. Do not check adjustType at the line or event level.
- if adjustType = "cancel_item", cancel the sales transaction whose number is given in the XML document. The sales transaction cannot have been cancelled previously. Do not check line or event adjustType.
- if adjustType = "adjust_item", look up the given sales transaction and attempt to adjust its associated readings or other header/item information, then verify if any adjustments are necessary at the line or event level. Any sales rep information included at this level will replace existing sales rep information for the item.
- To adjust a line or event of this sales transaction, but not the item itself, set adjustType = "no_item_action". The import server will look up the sales transaction item then go check the adjustType at the line level. No adjustment will be performed at the item level (this includes readings and sales reps associated with the item. Use "adjust_item" to adjust at the item level.

At the line level:

- if adjustType = "add_line", add the line to the current item. Add all events associated with this line without checking adjustType. Add sales reps and readings.
- if adjustType = "cancel_line", look up the specified line for the current item and cancel it. It cannot have been cancelled previously. Do not process events.
- if adjustType = "return_line", look up the specified line for the current item and return it. It cannot have been previously cancelled or returned.
- if adjustType = "adjust_line", look up the specified line for the current item and adjust it. Then check events associated with this line, if any exist. Any sales rep information included at this level will replace existing sales rep information for the line.
- if adjustType = "none", look up the specified line, then go check the line's events for adjustments.

Remember that if creating a whole new transaction, it is not necessary to specify adjustType at the line or event level.

At the event level:

- if adjustType = "add_event", add the event to the current line.
- if adjustType = "cancel_event", look up the specified event (by event type) for the current line, and cancel it. It cannot have been previously cancelled.

To adjust sales reps, include the new sales rep information with the item or line, and set adjustType to "adjust_item" or "adjust_line". The new sales rep information will replace the existing sales rep information for the item or line. Do not specify just the changes; give the whole new set of sales rep information.

In the vast majority of cases, the header level adjustType should be sufficient. It is rare that the customer can actually provide deltas to an existing sales transaction. It is much more likely that the entire sales transaction will be re-imported with an adjustment.

The data included in an adjusting XML transaction will replace existing data for the entity being replaced. This is important to remember, if you import only the header date, a new copy of the transaction header will be made with only the header date set.

If you adjust a line whose events have not yet been selected for crediting, the Sales Transaction Servicer will update the line "in place". If it has been credited, the servicer will create a new copy of the header or line and cancel the original header or line. This can result in significant difference in import performance.

The item type (the transType element in SalesTransItem) is required in the transaction XML because the item type indicates the transaction header profile AND the line profile. The import service needs the header profile and line profile so that it can verify that the correct header Custom Fields and line Custom Fields were included in the import. The line type, however, is not required in the transaction XML because it is considered simply an attribute of the line.

Dependent objects (must be present in the database before importing Sales Transaction data):

- Product
- Participant (sales rep code)
- Customer
- Customer Location (both ship-to and sold-to use this)
- Sales Transaction Type + Profiles and Custom Fields
- Sales Transaction Line Type + Profiles and Custom Fields
- Sales Transaction Event Type + Profiles and Custom Fields

Sales Transaction XML Schema (salesTrans.xsd)

The following elements (or fields) are defined in the XML Schema:

NOTE: Required fields in bold. Cardinality, or number of occurrences of the field in braces: {one or more}. No braces means that the field can occur only once. Attributes of an element are included in <>.

recordset: the root element of the input file.

- **SalesTransItem** <action> {one or more}

- transNumber (string)
- transType (string, validate against SalesTransactionType)
- customerID (string, = customer code, validate against Customer)
- shipToLocationID (string, = Location code, validate against Location)
- soldToLocationID (string, = Location code, validate against Location)
- description (string)
- transDate (date) valid date format = TBD)
- currency (string, = Currency code/mnemonic, validate against Currency)
- currencyConversionFactor
- SaleTransLine {one or more}
- lineNumber (string)
- lineDate (date)
- lineType (string, validate against SalesTransactionLineType)
- productID (string, = Product code, validate against Product)
- customField {zero or more}
- salesRep {zero or more}
- SalesTransEvent {one or more}
- eventType (string, validate against SalesTransactionEventType)
- eventDate (date)

- salesRep <salesRepID, rank, splitPercent> {zero or more} (salesRepID should be a valid Employee code)
- customField <readingType, numValue, stringValue> {zero or more} (readingType should be a valid ReadingType and belong to the profile associated with the ItemType or LineType)

Sample XML Document for salesTrans.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <SalesTransItem>

      <transNumber>101-232-X</transNumber>

      <transType>Invoice</transType>

      <customerID>101</customerID>

      <shipToLocationID>SFO</shipToLocationID>

      <soldToLocationID>SFO</soldToLocationID>
```

```
        <transDate>2001-11-04T10:00:00</transDate>
        <currency>USD</currency>
        <SalesTransLine>
        <lineNumber>1</lineNumber>
        <lineDate>2001-11-04T10:00:00</lineDate>
        <customField readingType="Qty" numericValue="1000"/>
        <customField readingType="UnitPrice" numericValue="10.00"/>
        <customField readingType="LeaseTerm" numericValue="36"/>
        <salesRep salesRepID="22237" rank="1"/>
        <salesRep salesRepID="23018" rank="2"/>
        <SalesTransEvent>
            <eventType>Ordered</eventType>
            <eventDate>2001-11-04T10:00:00</eventDate>
        </SalesTransEvent>
    </SalesTransLine>
<customField readingType="SalesAmount" numericValue="10000.00"/>
<customField readingType="ContractType" stringValue="Lease"/>
<salesRep salesRepID="22237" rank="1"/>
</SalesTransItem>
</recordset>
```

Credits (Also Known as Actuals or Performance Data)

The Credit object is not extensible, but it supports readings. Before importing Credits, you must design and set up the desired custom fields.

Core entities:

- Credit
- Credit Reading

Dependent entities:

- Measure (required)

- Profile (if importing readings)
- Custom Field (if importing readings)
- Participant (Employee, Customer, and/or Channel Partner)
- Organization
- Territory

A credit always has an associated Measure. The Measure has a Credit Profile (and a Goal Profile). The Credit Profile defines a group of Custom Fields. Before importing Credits, you must design the different Measures and their Profiles and Custom Fields, and create them in the UI.

A credit may be assigned to one and only one of the following: a Participant, a Territory, or an Organization. An import record containing references to a Participant and an Organization, for example, will be rejected.

Credit XML Schema (credit.xsd)

recordset: the root element of the input file.

- Credit {one or more}
  - measureID
  - participantID
  - organizationID
  - territoryID
  - description
  - earnedDate (string YYYY-MM-DD)
  - customField <name = "name" value = "value" currencyID="currencyID"> {zero or more}

The customFields (aka readings) must match the Custom Fields in the credit profile associated with the measure ID, or the record will be rejected. You can specify numeric or monetary values in the value attribute, just put quotes around it. Optionally indicate a currency if you like.

Sample Credit XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <Credit>

      <measureID>Bonus</measureID>

      <participantID>EMP101</participantID>

      <description>November credit for deal #123345</description>

      <earnedDate>2001-11-29</earnedDate>

      <creditReading name="Current Actual" value="5000.00"/>
```

```
        <creditReading name="Current Goal" value="10000.00"/>
        <creditReading name="YTD Actual" value="25000.00"/>
        <creditReading name="YTD Goal" value="20000.00"/>
    </Credit>
    <Credit>
        <measureID>Bonus</measureID>
        <participantID>EMP102</participantID>
        <description>November credits</description>
        <earnedDate>2001-11-23</earnedDate>
        <creditReading name="Current Actual" value="8000.00"/>
        <creditReading name="Current Goal" value="6000.00"/>
        <creditReading name="YTD Actual" value="45000.00"/>
        <creditReading name="YTD Goal" value="20000.00"/>
    </Credit>
</recordset>
```

Goals

The Goal object supports readings. Before importing Goals, you must design and setup the desired Custom Fields, Profiles, and Measures.

Core entites:

- Goal
- Goal Reading

Dependent entities:

- Measure (required)
- Profile
- Custom Field
- Participant (Employee, Customer, or Channel Partner)
- Organization
- Territory

A Goal always has an associated Measure. The Measure has a Credit Profile and a Goal Profile. The Goal Profile defines a group of Custom Fields. Before importing Goals, you must the different Measures and their Profiles and Custom Fields, and create them in the UI.

A Goal may be assigned to one and only one of the following: a Participant, a Territory, or an Organization. An import record containing more than one of the three will be rejected.

Goal XML Schema (goal.xsd)

recordset: the root element of the input file.

- Goal (one or more)
- measureID
- participantID
- organizationID
- territoryID
- description
- goalReading name= "readingTypeCode" value="goalValue" currencyID="EUR" {zero or more}

The customFields (aka readings) must match the Custom Fields in the goal profile associated with the measure ID, or the record will be rejected. You can specify numeric or monetary values in the value attribute, just put quotes around it. Optionally indicate a currency if you like.

Sample Goal XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset >

   <Goal>

      <measureID>Revenue</measureID>

      <participantID>E001</participantID>

      <description>October Goal for Sales Reps</description>

      <goalDate>2002-10-01</goalDate>

      <goalReading name="SalesAmount" value="100000" currencyID="USD"/>

      <goalReading name="UnitsSold" value="2000"/>

   </Goal>

   <Goal>

      <measureID>NewAccounts</measureID>

      <orgID>BRANCH_001</orgID>
```

```
        <description>October New Accounts Goal for Branch 001</description>
        <goalDate>2002-10-01</goalDate>
        <goalReading name="NewAccounts" value="200"/>
    </Goal>
</recordset>
```

Product and Product Hierarchy

The Product object supports Extensible Attributes. A product may have zero or more Product Measures (many-to-many association with a Measure object). Each Product Measure has a Product Measure Name. A product may have zero or more Product Rates (each with a Rate).

Core entities:

- Product
- ProductMeasure
- ProductRate

Dependent entities:

CAUTION: These entities must be set up in the database before importing product data.

- RateGroup (if importing Product Rate information)
- Measure (if importing ProductMeasure information)
- Extended Attributes for the Product object

Product Hierarchy import is not fully supported at this time.

Product XML Schema (product.xsd)

All datatypes are String unless specified otherwise

recordset: the root element of the input file.

- Product {one or more}
  - Code
  - name
  - modifier
  - unitCost (float)
  - unitPrice (float)
  - unitMargin (float)
  - unitOfMeasure
  - defaultRate

- productType
- description
- hierarchyLevelID (natural key of a Hierarchy Level object)
- parentProductID (natural key of a Product object)
- ProductMeasure <measureID = "measure" productMeasureName="name"> {zero or more}
- ProductRate <rateGroupID = "rateGroupID" rate="rate"> {zero or more}
- extAttribute <name="name" value="value">{zero or more}

Sample XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <Product>

      <code>MW502PF</code>

      <name>Magic Wand Model 501 (Phoenix Feather core)</name>

      <modifier>0.35</modifier>

      <unitCost>100.00</unitCost>

      <unitPrice>250.00</unitPrice>

      <unitMargin>250</unitMargin>

      <unitOfMeasure>EA</unitOfMeasure>

      <defaultRate>.1</defaultRate>

      <productType>magic wand</productType>

      <description>13 1/2 inch magic wand with phoenix feather and dragon scales</
      description>

      <hierarchyLevelID>Product</hierarchyLevelID>

      <parentProductID>MW5x</parentProductID>

      <ProductMeasure measureID="Revenue" productMeasureName="WandRevenue"/>

      <ProductMeasure measureID="MagicItems" productMeasureName="MagicItemCount"/>

      <ProductRate rateGroupID="MagicWand" rate=".1"/>

      <extAttribute name="coreMaterial" value="phoenix feather"/>

   </Product>

</recordset>
```

Customer (Also Known as Account)

The Customer object supports Extensible Attributes. It contains two child entities: CustomerLocation and CustomerContact. One customer record may contain several of each of these entities. Each CustomerLocation must have a unique code for the Customer it belongs to. Two Customers may have CustomerLocations with the same codes, however.

Core entities:

- customer

Dependent entities:

- Employee (if accountMgrID or nationalAccountRepID is imported)
- ChannelSegment
- Territory
- Currency
- Customer (if parentCustomerID is imported)

If importing parentCustomer, the parent customer record must already exist in the database.

A comment on circular dependencies:

If you are importing customer with customer location records, the customer.shipToLocationID and customer.soldToLocationID fields must be null. To set these fields, you need to create a second import file containing the customer code and the shipTo- and soldToLocationID values, and the import service will update the references for you. The reason for importing this information in two passes is that the Customer object must be created first, then the CustomerLocation object can be created. However the customer.shipToLocationID and customer.soldToLocationID fields are both foreign keys to the Customer Location object. You cannot create a foreign key reference to an object that does not yet exist in the database. Therefore, those two fields must be null when creating the Customer object. You can update the Customer object in a second pass with shipTo and soldTo information.

Note that these two fields need not be populated unless your crediting rules depend upon the existence of this information.

CustomerLocation codes: for any given customer, no two customer locations may have the same code. Same goes for CustomerContact.

You cannot include the same customer location code twice in an import record for a given customer. This will cause the Import Service to hang.

Customer XML Schema (customer.xsd)

recordset: the root element of the input file.

- Customer {one or more}
  - code
  - name

- participantFlag (boolean default = false)
- lastName
- firstName
- middleName
- address1
- address2
- city
- state
- zip
- country
- officePhone
- officeFax
- email
- URL
- SSN
- TaxID
- industryCode
- DandBNumber
- annualRevenue
- timeZone
- customerType
- customerLevel
- customerStatus
- individualFlag (boolean default = false)
- payEligibiligyFlag (boolean default = false)
- goodCreditFlag (boolean default = false)
- beginDate (string YYYY-MM-DD)
- endDate (String YYYY-MM-DD)
- rank (integer)
- leadSourceType
- leadSource
- parentCustomerID (natural key of Customer object)
- accountMgrID (natural key of Employee object)

- nationalAccountRepID (natural key of Employee object)
- channelSegmentID (natural key of ChannelSegment object)
- territoryID (natural key of Territory object)
- shipToLocationID (natural key of a CustomerLocation object)
- soldToLocationID(natural key of a CustomerLocation object)
- currencyID (natural key of Currency object)
- noCapFlag (boolean, default = false)
- CustomerLocation (zero or more)

- Customer Location (zero or more)
  - code
  - name
  - businessPurpose
  - currencyID (natural key of Currency object)
  - address1
  - address2
  - city
  - county
  - state
  - zip
  - country
  - officePhone
  - officeFax
  - extAttribute <name="name" value="value> {zero or more}
  - CustomerContact (zero or more)
  - code
  - lastName
  - firstName
  - middleName
  - title
  - primaryFlag (true | false, default = false)
  - address1
  - address2
  - city

- county
- state
- zip
- country
- officePhone
- officeFax
- email
- URL
- role
- extAttribute <name="name" value="value> {zero or more}
- extAttribute <name = "name" value="value> {zero or more}

Sample Customer XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <Customer>

      <code>SGI</code>

      <name>Silicon Graphics, Inc</name>

      <participantFlag>true</participantFlag>

      <address1>1600 Amphitheatre Parkway</address1>

      <city>Mountain View</city>

      <state>CA</state>

      <zip>94043</zip>

      <country>USA</country>

      <email>info@sgi.com</email>

      <TaxID>92-12345-67</TaxID>

      <timeZone>PST</timeZone>

      <individualFlag>true</individualFlag>

      <CustomerContact>

         <code>CC001</code>

         <lastName>Took</lastName>
```

```
            <firstName>Peregrin</firstName>
            <primaryFlag>true</primaryFlag>
            <email>ptook@mycontact.com</email>
            <URL>www.mycontact.com</URL>
            <role>General Manager</role>
        </CustomerContact>
        <CustomerContact>
            <code>String</code>
            <lastName>Brandybuck</lastName>
            <firstName>Meriadoc</firstName>
            <email>mbrandybuck@mycontact.com</email>
            <URL>www.mycontact.com</URL>
            <role>Technical Engineer</role>
        </CustomerContact>
        <CustomerLocation>
            <code>CL001</code>
        </CustomerLocation>
        <CustomerLocation>
            <code>CL002</code>
        </CustomerLocation>
        <extAttribute name="foo" value="1"/>
        <extAttribute name="bar" value="a"/>
    </Customer>
</recordset>
```

Organization and Organization Hierarchy

The Organization object is extensible, but importing extensible attributes is not yet implemented. Importing an Organization Hierarchy also is not yet supported. It is possible to import the organization's level in the hierarchy though.

NOTE: There is a circular dependency between organization and employee. Each table refers to the other. If you import organizations first, you cannot include the organization manager Employee ID, since it does not yet exist. If you import employees first, you cannot include the organization ID in the employee Job structure, since it does not yet exist. You have to create a third import to update employee or organization with the relevant foreign key information.

Dependent Objects:

- Hierarchy Level
- Location
- Cost Center
- Currency
- Employee (manager of the organization)

Organization Schema (organization.xsd)

recordset: the root element of the input file.

- Organization {one or more}
  - code
  - name
  - modifier
  - description
  - locationID (natural key of a Location object)
  - costCenterID (natural key of a CostCenter object)
  - hierarchyLevelID (natural key of a Hierarchy Level object)
  - currencyID (natural key of a Currency object)
  - managerEmployeeID (natural key of an Employee object)

Sample XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <Organization>
```

```
        <code>ORG1</code>
        <name>Organization 1</name>
        <modifier>1.5</modifier>
        <description>A description of an Organization</description>
        <locationID>LOC1</locationID>
        <costCenterID>CC1</costCenterID>
        <hierarchyLevelID>Branch</hierarchyLevelID>
        <currencyID>USD</currencyID>
        <managerEmployeeID>EMP101</managerEmployeeID>
    </Organization>
</recordset>
```

Territory and Territory Hierarchy

Territory is probably one of the most complex entities that you can import, if you take advantage of the powerful qualifier mechanism. You can define extended attributes on the Territory object itself.

The Territory Hierarchy is made up of two types of Territory objects: Regions and Territories. These types are different from hierarchy levels. Design your territory hierarchy levels carefully to avoid confusion. The "leaf" nodes of the hierarchy MUST be Territory-type territories. A leaf node has no child nodes. All other non-leaf nodes MUST be Region-type territories. Only Territory-type territories can have qualifiers. Region-type territories may not have qualifiers but they may have assigned participants.

Territory Qualifier Conditions are expressions that are evaluated at time of crediting to determine which territory(s) or participants assigned to territories should receive credit for a transaction. Siebel defines a set of qualifier attributes that you can use to create complex, powerful expressions. These attributes are:

- product
- customer
- city
- county
- state
- province
- zip
- country
- areaCode

For example, you can define a territory qualifer condition to be this expression:

"zip between 64000 and 54999" AND "customer = 'ACME'"

Territories can belong to Channel Segments so you can define vertical territory hierarchies in the same geographical space. You must create Channel Segments before you can import territories related to them.

New in 9.0: Territory Qualifier Conditions can be built using transaction line or header readings (a.k.a custom fields), allowing expressions like the following:

salesAmount >= 10,000" where salesAmount is a custom field defined in the transaction line or header profile.

A word on updating territories: the Territory Import Service does not perform incremental updates. That is to say, it cannot compare the contents of the import record with the existing qualifiers, conditions, and assignments to see what changed. It simply removes all existing qualifiers, conditions, and assignments, and creates new qualifiers, conditions, and assignments from the data contained in the import record. In other words, the Territory Import Service will always completely replace the existing territory record with the contents of the import XML.

The ManagerParticipantID is different from the Participant assigned to the Territory.

Core entities:

- Territory
- Territory Assignment
  - Links a territory with a participant.
  - More than one participant can be assigned to a territory.
  - Participants can have Roles in a territory
- Territory Qualifier (a container for Territory Qualifier Conditions)
  - Territory Qualifier Condition

Dependent entities (must be created before you can import territories):

- Territory Role

  This is the list of roles that participants can have in a territory
- Hierarchy Levels
- Participants
- Channel Segment

Territory Schema (territory.xsd)

recordset: the root element of the input file.

- Territory
  - code
  - name

- type ("region" or "territory")
- description
- hierarchyLevelID (natural key of a Hierarchy Level)
- status (freeform field for your use. no meaning to Siebel)
- managerParticipantID (natural key of a Participant object)
- channelSegmentID
- currencyID
- parentTerrID (if this is the root node in the hierarchy, set = "root")
- assignment (zero or more)

- assignment (zero or more
  - participantID
  - role
  - startDate (YYYY-MM-DD)
  - endDate(YYYY-MM-DD)
    - qualifier (zero or more. Only allowed if type="territory")
  - condition <type="attribute|reading">
    - terrAttributeCode
    - salesTransContext ("line" or "header")
    - readingTypeCode
    - operator
    - value1
    - value2 (only if operator = "between")
- extAttribute <name="name" value="value">

Here is a more detailed description of the schema. The <condition> element which belongs to the <qualifier> element must include one attribute called <type>. The allowable values of this <type> element are "reading" and "attribute". The "reading" type condition means that this condition is based on a reading from a sales transaction line or header. The <salesTransContext> element will indicate whether the reading will be from the sales transaction line or the sales transaction header.

If the <condition> type attribute's value is "attribute", the condition is based on one of the Territory Qualifier Attributes described above, like zip or product or customer. In this case, salesTransContext and readingTypeCode may be omitted from the XML document for this condition.

If the <condition> type is "reading", the condition will be based on the readingTypeCode and salesTransContext. Both these elements must be included in the XML. The import service will create a new Territory Qualifier Attribute if it doesn't exist for the given reading type.

The allowable values of <operator> are case sensitive. They are as follows:

- equal
- notEqual
- like
- notLike
- inList
- notInList
- greaterThan
- greaterThanOrEqual
- lessThan
- lessThanOrEqual
- between
- notBetween

If using the "like" and "notLike" operator, <value1> may include a Perl5-style regular expression. For more information on Perl 5 regular expressions, please see www.perl.com or pick up a copy of Programming Perl from O'Reilly.

If using the inList or notInList operator, <value1> should be a string of comma-separated values, like this: "a, b, c, z".

If using the "between" or "notBetween" operator, you must include <value1> and <value2>. All other operators ignore <value2>.

Sample Territory XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <Territory>

      <Code>D995</code>

      <name>Direct 995</name>

      <type>territory</type>

      <description>Direct 995</description>

      <hierarchyLevelID>Territory</hierarchyLevelID>

      <channelSegmentID>Direct</channelSegmentID>

      <parentTerrID>335</parentTerrID>

      <assignment>
```

```
        <participantID>E001</participantID>
        <role>Sales Representative</role>
    </assignment>
    <qualifier>
        <condition type="attribute">
        <terrAttributeCode>zip</terrAttributeCode>
        <operator>between</operator>
        <value1>99500</value1>
        <value2>99599</value2>
    </condition>
    </qualifier>
</Territory>
<Territory>
    <code>D968</code>
    <name>Direct 968</name>
    <type>territory</type>
    <description>Direct 968</description>
    <hierarchyLevelID>Territory</hierarchyLevelID>
    <channelSegmentID>Direct</channelSegmentID>
    <parentTerrID>330</parentTerrID>
    <assignment>
        <participantID>E002</participantID>
        <role>Sales Representative</role>
    </assignment>
    <qualifier>
        <condition type="attribute">
        <terrAttributeCode>zip</terrAttributeCode>
        <salestransContext>header</salestransContext>
        <operator>between</operator>
        <value1>96800</value1>
```

```
            <value2>96899</value2>
        </condition>
        <condition type="reading">
            <readingTypeCode>salesAmount</ readingTypeCode >
            <salestransContext>header</salestransContext>
            <operator>greaterThan</operator>
            <value1>10000</value1>
        </condition>
    </qualifier>
</Territory>
</recordset>
```

The first <Territory> record will create a Territory definition with a condition based on the zip code. The second <Territory> record will create conditions based on zip code and salesAmount, a reading type that will be expected on the header profile at time of territory qualification evaluation.

Channel Partner

The Channel Partner entity is extensible.

Channel Partners are participants. Make sure to specify <participantFlag>true</participantFlag> in order for the import service to create channel partners as participants eligible for plans.

Dependent Entities

- Employee (if you are linking channel partners to Employees using the channelManagerID field)
- ChannelSegment (if you are linking channel partners with channel segments)

Channel Partner Schema (channelPartner.xsd)

recordset: the root element of the input file.

- ChannelPartner
  - code
  - name
  - participantFlag (default = false)
  - address1
  - address2
  - city

- state
- county
- zip
- country
- officePhone
- officeFax
- email
- URL
- taxNumber
- timeZone
- level
- status
- payEligibleFlag (default = false)
- beginDate
- endDate
- rank
- parentChannelPartnerID (natural key of a Channel Partner)
- channelManagerID (natural key of an Employee)
- channelSegmentID
- currencyID
- channelContact (zero or more)

- channelContact (zero or more)
  - contactCode
  - lastName
  - firstName
  - middleName
  - title
  - primaryFlag (true or false, default = false)
  - address1
  - address2
  - city
  - state
  - county

- zip
- country
- officePhone
- officeFax
- email
- URL
- role

- role
  - channelCert (zero or more)
  - certCode
  - certName
  - level
  - certifiedUsers
  - rank
- extAttribute <name="name" value="value">

Sample Channel Partner XML Document

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <ChannelPartner>

      <code>CP1</code>

      <name>Channel Partner 1</name>

      <participantFlag>true</participantFlag>

      <address1>123 Elm Street</address1>

      <city>Pleasanton</city>

      <state>CA</state>

      <zip>94566</zip>

      <officePhone>+1(800)555-1234</officePhone>

      <email>cp@somecompany.com</email>

      <URL>www.somecompany.com</URL>

      <taxNumber>32-1234567-8</taxNumber>

      <status>ACTIVE</status>
```

```
        <payEligibleFlag>true</payEligibleFlag>
        <channelManagerID>E001</channelManagerID>
        <channelSegmentID>Direct</channelSegmentID>
        <currencyID>USD</currencyID>
        <channelContact>
            <contactCode>CON1</contactCode>
            <lastName>Smith</lastName>
            <firstName>Fred</firstName>
            <title>Global Operations Manager</title>
        </channelContact>
            <channelCert>
            <certCode>CC1</certCode>
            <certName>Channel Cert 1</certName>
            <level>A</level>
            <certifiedUsers>50</certifiedUsers>
            <rank>1</rank>
        </channelCert>
        <channelCert>
            <certCode>CC2</certCode>
            <certName>Channel Cert 2</certName>
            <level>B</level>
            <certifiedUsers>80</certifiedUsers>
            <rank>2</rank>
        </channelCert>
        <extAttribute name="attribute_1" value="A"/>
    </ChannelPartner>
</recordset>
```

Exchange Table/Rate

The exchange table and exchange rate is not extensible, nor does it support profiles and readings.

You need to specify exchange rates in both directions; the import will not perform any rate calculations. In other words, if I tried to import USD->HKD = "1.5", the system would not add HKD->USD=.66 for me.

Dependent objects:

- Currency

Exchange Table Schema (exchange.xsd)

recordset: the root element of the input file.

- exchangeTable
  - code
  - name
  - date
  - <exchangeRate fromCurrencyID="XXX" toCurrencyID="YYY" rate="1.5"/> {one or more}

Sample XML Document

```
<?xml version="1.0" encoding="UTF-8"?>

<recordset>

   <exchangeTable>

      <code>ET2001_12_21</code>

      <name>Exchange Table for Dec 21, 2001</name>

      <date>2001-12-21</date>

      <exchangeRate fromCurrencyID="USD" toCurrencyID="EUR" rate="1.12716"/>

      <exchangeRate fromCurrencyID="EUR" toCurrencyID="USD" rate="0.88718"/>

   </exchangeTable>

</recordset>
```

Matrix Expression

Matrix Expressions are not extensible nor do they support readings.

row and column sizes - specify the number of data rows and data columns. The system will account for the row header and the column header.

Sample Matrix Variation (in Tabular Format)

This is the Origin Cell			
MyRowLabel	**MyColumnLabel: A**	**MyColumnLabel: B**	**MyColumnLabel: C**
0-10000	1	2	3
10001-15000	4	5	6
15001-999999	7	8	9

Attributes of this matrix variation:

- matrix type = "rows_cols"
- row count = 3
- column count = 3
- row label = "MyRowLabel"
- row header type = "numbers_range"
- column label = "MyColumnLabel"
- column header type = "string"
- cells: there are 3 row header cells, 3 column header cells, and 9 data cells, for a total of 15 cells. The import service will create the origin cell for you.

Business Rules

- A matrix expression must always have a "default" variation. In addition to a default variation, a matrix expression may define versions or variations (aka dimensions) for specific job codes, organization codes, or participant codes.
- If the matrix type is 'rows_cols", you must specify column label and column header type.

Rounding Rules

If you create a matrix through the UI, you can specify a rounding rule to be used for column and for row entries. This feature is not yet supported by the import service. You can include rounding rule codes in the import XML but they will be ignored by the import service.

XML Schema

recordset: the root element of the input file.

- matrix <code = "code" name = "name" [currencyID="USD"]>{one or more}
  - variation <variationType=default|job|org|participant rowCount="N" colCount = "N">{one or more}
    - orgID

- participantID
- jobID
- description
- matrixType values: rows|rows_cols
- rowLabel
- rowHeaderType values: string|string_range|string_wildcard|number |number_range |date | date_range
- colLabel
- colHeaderType (see values of rowHeaderType)
- cell <row = "N" col = "N" headerFromValue="A" headerToValue="B" resultValue="999.99">

General Validations

NOTE: These general validations are applied to all cases of matrix create/update.

1. Ifexpression variation = Default, org/job/participant codes in the import record will be ignored
2. Ifexpression variation = Org/Job/Participant, the matrix variation will be rejected unless
3. The given org/job/participant code exists for the operating unit.
4. Missing cell values will be set to 0.0.
5. col and row rounding rule codes must correspond to rounding rules valid in the OU.
6. The (0,0) cell will be created automatically. any data passed in for this cell will be ignored.
7. Date or date range row and column headers should be in the form MM/DD/YYYY.
8. Date values will be parsed into Date objects and stored as Date.getTime(). Invalid dates will cause the entire matrix variation to be rejected.
9. If matrix type (apply to type) is "rows", these elements will be ignored: colCount, colHeaderType, colLabel, colRoundingRuleID
10. If matrix type is "rows_cols", the import record will be rejected unless colCount, colHeaderType, colLabel are present in the record. colRoundingRuleID is optional.

Sample XML document

NOTE: See matrix.xsd for schema

```
<matrix code="myMatrix" name="My cool Matrix" currency="USD" description="The most excellent Matrix">

   <variation variationType="default" rowCount="3" colCount="3">

      <matrixType>rows_cols</matrixType>

      <rowLabel>Branch ID</rowLabel>
```

```
        <rowHeaderType>string</rowHeaderType>
        <rowRoundingRuleID>None</rowRoundingRuleID>
        <columnLabel>FeeOverage</columnLabel>
        <colHeaderType>number</colHeaderType>
        <colRoundingRuleID>None</colRoundingRuleID>
        <data>
            <cell row="1" col="0" headerFromValue="NW"/>
            <cell row="2" col="0" headerFromValue ="NE"/>
            <cell row="3" col="0" headerFromValue ="SW"/>
            <cell row="0" col="1" headerFromValue ="1"/>
            <cell row="0" col="2" headerFromValue ="2"/>
            <cell row="0" col="3" headerFromValue ="3"/>
            <cell row="1" col="1" resultValue="1"/>
            <cell row="2" col="1  resultValue="2"/>
            <cell row="3" col="1" resultValue="3"/>
            <cell row="1" col="2" resultValue="4"/>
            <cell row="2" col="2" resultValue="5"/>
            <cell row="3" col="2" resultValue="6"/>
            <cell row="1" col="3" resultValue="7"/>
            <cell row="2" col="3" resultValue="8"/>
            <cell row="3" col="3" resultValue="9"/>
        </data>
    </variation>
</matrix>
```

Matrix XML creation notes

- Use the "headerFromValue" and "headerToValue" attributes of the <cell> element to specify row and column header values.
- Use headerToValue to specify the upper bound of a range, or leave null if row|col header type is not a range.

- Use the "resultValue" attribute of the <cell> element to specify result values. Even though resultValue will be quoted since it is an attribute, the value will be parsed to a number (a double primitive, to be precise).
- The XML schema contains enumerations for all appropriate attributes and elements. Please validate your XML document against the schema to insure it will be imported correctly.
- You must always include a matrix variationType="default" when creating a new matrix. If you are importing new variations for an existing matrix, you need not include a default.

In general, for a cell representing a column header or row header, always use the "headerFromValue" attribute and only use "headerToValue" if the row header type or column header type is one of the three range types: number_range, string_range, or date_range.

File Adaptor

Each Siebel Incentive Compensation Management installation includes a command line utility called "File Adaptor" that can be used to translate import files generated by another system into the XML format that Siebel Incentive Compensation Management recognizes. The File Adaptor can be accessed on any type of server (Windows, Solaris, or AIX) using a command line window.

The File Adaptor utility can be found at INSTALL.DIR\tools.

The general structure of the command line is as follows:

```
convert [-d] -if inputFile  [-mf mappingFile]  [-of outputFile] [-xf XSLStylesheet]
```

- **-d** turns on the debugging function for the conversion process. This is optional and does not need to be included every time the utility is run.
- **inputFile** tells the utility where to find the source import file. Along with the file name, include the relative or absolute path to the file if it is not located in the same directory as the File Converter utility.
- **mappingFile** tells the converter which mapping file to use to perform the intermediate XML transformation. If no value is specified here, the system will automatically choose a standard mapping file.
- **outputFile** tells the converter which file will hold the mapped results from the intermediate mapping transformation. If no value is specified, the output will be written to the STDOUT file.
- **XSLStyleSheet** instructs the utility to use the specified XSL style sheet to transform the intermediate values in the output file into final output values. If no value is specified here, the system assumes that the values in the output file are the final output values and uses those.

In each case, the brackets are not included in the actual command line; they simply denote parts of the line that can be left out if you want or need to use the default options for those parts.

As an example, the following command line gets an employee data import file (employee.txt) and transforms it according to an intermediate mapping file and an XSL style sheet expressly set up for employee data:

```
convert -if employee.txt -mf employee_map.xml -of employee_output.xml -xf employee.xsl
```

For the File Adaptor utility to work successfully, the input file must be a delimited plain text file (*not* a fixed width text file). This means that the source application that generates the import file must be able to export data in delimited plain text format, which most applications are capable of.

18 Siebel CRM to Siebel ICM Integration

Implementations in which Siebel CRM applications and the Siebel Incentive Compensation Management application with co-exist require handling of overlapping data entities that are present in both applications. In such cases, Siebel CRM should be considered the system of record with respect to common reference data, including, but no limited to, Employees, Products, and Jobs. This data will be extracted from Siebel CRM to synchronize with Siebel ICM. Changes made to Siebel ICM data will not be copied or loaded back into Siebel CRM.

A set of reference implementations of this loosely coupled integration approach are available in Siebel ICM and do not require any special or add-on product components for Siebel CRM beyond the configuration steps described in this chapter.

Siebel Server Configuration

You must have a working Siebel Server. The following components must be installed:

Workflow "Incentive Comp Extract Workflow"

To import the Incentive Comp Extract Workflow

1. Start a Siebel Client application and connect to your Siebel Server.
2. Navigate to Business Process Administration screen > Workflow Processes view.
3. Locate the file in the Siebel source tree or distribution build under the directory etc/SiebelTools/Workflows called "Incentive Comp Extract Workflow.xml".
4. From the applet menu, choose Import Workflows and browse to the workflow .xml file.
5. After the workflow is imported, make sure it is activated.

Configuration Properties

Certain properties must be set in order for the interface to be able to connect to a Siebel Server and extract data. These properties are set in the main properties files:

Development Build

Set the following properties in masterBuild.properties:

```
siebel.connectString=[yourConnectString]

siebel.language=ENU

siebel.username=[siebelUsername]
```

```
siebel.password=[siebelPassword]
```

where property "siebel.connectString" is used to connect to an instance of the Siebel Server. An example of the connect string looks something like this:

```
siebel.connectString= siebel.TCPIP.none.NONE://myserver:2320/siebel/XXXObjMgr_enu/
siebelServer
```

where myserver is usually the hostname of the machine running the instance. siebelServer is also usually the hostname of the machine. XXXObjMgr_enu is the name of the Application Object Manager you will connect to. For example, an instance configured to run FINS will have an application object manager name of "FINSObjMgr_enu".

Distribution Build

Set the following properties in install.properties:

```
siebel.connectString=[yourConnectString]

siebel.language=ENU

siebel.username=[siebelUsername]

siebel.password=[siebelPassword]
```

then proceed with running setup.

Interface Configuration

The interface itself may be configured to extract certain entities and apply desired transformations. No configuration is required out of the box, unless you wish to extract different integration objects, use a custom workflow, or apply your own transformations to the extracted data.

To modify the interface configuration, locate the file "siebelInterfaceConfig.xml.in" and open it in an editor. It should be in the "config" directory. IMPORTANT: make sure you edit the file ending in ".in". If you edit the file with the ".xml" extension, your changes will be wiped out the next time you do a build.

Here is a fragment of "siebelInterfaceConfig.xml.in":

```
<?xml version="1.0" encoding="UTF-8"?>

<siebelInterfaceConfig

   name="reference"

   connectString="@siebel.connectString@"

   langCode="@siebel.language@"

   siebelUsername="@siebel.username@"

   siebelPassword="@siebel.password@">

      <!-- extract Employee data -->
```

```
<!-- change integrationObjectName to name of desired integration object -->
<workflowConfig name="myWorkflow"
   siebelWorkflowName="Incentive Comp Extract Workflow"
   integrationObjectName="EAI Employee"
   dumpToFile="c:/temp/employee.xml">
<searchSpec></searchSpec>
<!--   sample search spec
<searchSpec><![CDATA[[Employee.Login Name] like "A*"]]></searchSpec>
-->

<!-- add additional transform elements if you wish to transform the extract output
additional ways -->
<transform addToExportSet="true"
   stylesheet="employee.xsl"
   entityClassName="com..ce.employee.EmployeeVersionImpl"/>
</workflowConfig>
<workflowConfig>   … additional invocations of the workflow …
```

Let's examine each element here:

1. siebelInterfaceConfig: this is the root element. It has 5 attributes:
   a. name: this is the name of the interface configuration. You can give it any value you like.
   b. connectString - don't change this. It will be set during the build
   c. langCode - don't change. Will be set by build.
   d. siebelUsername: - don't change. Will be set by build.
   e. siebelPassword: - don't change. Will be set by build.
2. workflowConfig: this element will invoke the workflow on the Siebel side one time. It has three attributes:
   a. name [required]: this is a descriptive name with no other significance.
   b. siebelWorkflowName [required]: this is the name of the Siebel workflow that will be invoked. The default value "Incentive Comp Extract Workflow", should suffice, unless you have built and activated a custom workflow. See section above for instructions on importing the workflow definition into Siebel. The workflow definition is included in the Siebel distribution.

c integrationObjectName [required]: this is the name of the integration object which the workflow will extract. The name should correspond to an integration object present in the Siebel Server repository. If you have created a custom integration object please ensure it has been compiled into the repository.

d dumpToFile [optional]: The value of this attribute should be the filename to dump the raw data to. This is useful if debugging the search specification or in preparation for writing a new stylesheet.

The "workflowConfig" element contains two child elements:

1 searchSpec: the text of this element is a search specification string that will be passed to the query that will extract data for the integrationObjectName given above. The *Siebel Bookshelf* documents the syntax of the search specification string. NOTE: Only 1 searchSpec element may be present.

2 transform: has 4 attributes. May be present one or more times. Each "transform" element represents a single transformation that will be run against the data extracted by the workflow. Attributes include:

- addToExportSet: this is a Boolean flag that indicates whether the output of the transformation should be part of an export set or not. Set to "true" to add the transform output to an export set. If true, the "entityClassName" attribute must be set.
- entityClassName: the entity class name to create export set entry for. This attribute should have the value of the entity class you are creating, for example, com.motiva.ce.employee.EmployeeVersionImpl. See masterEntityExportList.properties for the list of valid entityClassNames. It MUST be set if addToExportSet="true".
- outputFilename: set this to the filename you wish output XML written to. Use this attribute if you are NOT generating XML to be part of an export set. The filename should be fully qualified appropriate for your platform, otherwise it will be written to the current working directory of the application.
- stylesheet: the name of the stylesheet, e.g. "employee.xsl". The stylesheet must be located in the directory <jboss_root>/migration/conf/xsl/siebel, or the transformation will fail. You do not need to include any path information unless you have created subdirectories under conf/migration/xsl/siebel.

  In addition to the reference implementation stylesheets, an "identity" transformation stylesheet has been included, called "identity.xsl". You can use this if you just wish to see the resulting output of the extract. It is subtly different from the "dumpToFile" attribute above - dumpToFile writes the raw data from Siebel, where the identity transformation writes the data after it has been run through the transformation engine (xalan).

You can have multiple workflowConfig elements and multiple transform elements within one workflowConfig element.

Running the Interface

Running the interface from Windows (server also on Windows)

1. Make sure the Siebel Server is running.
2. Bounce the Siebel Server service if necessary.
3. Open a command window.
4. Make sure ANT_HOME and JAVA_HOME are set.
5. Run the following command: ant siebelExtract.
6. Locate the export set created by the extract and run Operating Unit Import.

   Alternatively, import each file separately (if outputFilename and addToExportSet=false).

19 Operating Unit Export and Import

Siebel Incentive Compensation Management provides a built in utility to perform an Operating Unit export or import. This feature will be helpful when moving data between a production environment and a test or development environment. The Operating Unit export process creates a .zip file that contains xml files for all of the following objects:

- Extended attributes
- Profiles
- Context Attributes (excluding System-owned Context Attributes)
- Templates (Condition and Recipient, excluding System-owned)
- Organizations
- Organization hierarchy
- Products
- Product hierarchy
- Measures
- Employee participants
- Custom fields
- Frequency groups
- Draw/cap profiles
- Plan type
- Incentive types
- Transaction types
- Transaction line types
- Transaction event types
- Cost center
- Channel segment
- Rate group
- Job code
- Rounding rule
- Payroll system
- Location
- Salary grade

- Credit rule sets
- Step calculations
- Matrix calculations
- Threshold calculations
- Formulas
- Plans

Exporting Operating Unit Sets

To export operating unit sets

1. Click Services on the Main Navigation bar, then select All Services.
2. Select Operating Unit Export from the Service Type drop-down list.
3. Enter a comment about this export in the Comment field.
4. Select a log level from the Log Level drop down list.
5. Click Launch Service.
6. Click Refresh in the Exports section of the service manager to verify that the service has completed successfully.
7. Click Services on the Main Navigation bar, then select Import & Export Sets.
8. Click the Download button to download this export set to any location you designate.
9. Click Save.
10. Select the location where you want to save the zip files from the Save in drop-down list.
11. Rename the file in the File Name field (optional).
12. Click Save.

Viewing Exported Files

Once the .zip file has been saved to a designated location, you can move it to another machine, review the files, or edit the files.

NOTE: It is not recommended that the xml files be edited directly.

To view exported files

1. Double-click on the zip file you downloaded.

   NOTE: Winzip must be installed on your machine to perform an extraction of the files.

2. Double-click on one of the files in the winzip archive.

3 The XML for that file is displayed.

4 Close the .xml file.

Uploading Operating Unit Sets

Once you have moved the zip file to another machine or made the necessary edits to these files you can upload the export set and import the files back into Siebel Incentive Compensation Management.

NOTE: These files must all be in the .zip file when you import them back into the system. If a file is removed the import will fail.

To upload operating unit sets

1 Click Services on the Main Navigation bar, then select Import & Export Sets.

2 Click the Browse button next to the Import File field to locate the export set you want to import.

3 Navigate to the .zip file.

4 Click Open.

5 Click Upload.

NOTE: When you upload the files to the same machine that they were downloaded from, the system will simply over-write the existing file with the newly uploaded one. If you have uploaded these files to a new machine, the upload will display in the export set screen.

Importing Operating Unit Sets

Once you have uploaded the files to the appropriate machine you must import these files to access the data contained in the operating unit.

To import operating unit sets

1 Click Services on the Main Navigation bar, then select All Services.

2 Select Operating Unit Import from the Service Type drop-down list.

3 Click the search button to select the operating unit you want to import into this scenario.

4 Click Launch Service.

5 Click the Refresh button in the Import section of the screen to verify that the import has completed successfully.

NOTE: The working period that you are in when you import the operating unit will become the first effective working period for that data.

20 Customizing Dashboards

Siebel Incentive Compensation Management's dashboards are displayed using Jetspeed portlets. In most cases, the dashboards will be used in their default configuration; however, if customization is required, follow the instructions below. Customization of portlets is limited to which registered portlet will appear in a content pane for a given dashboard. Documentation specific to Jetspeed's portal customization interface can be found at http://jakarta.apache.org/jetspeed/site/customizer.html. See "About Portlet Registration" if you wish to register new portlets to make them available within the dashboards for customization.

NOTE: For general navigation assistance and information throughout this document, please refer to the Jetspeed user documentation at: http://jakarta.apache.org/jetspeed/site/index.html.

NOTE: For information on creating your own custom portlet class, refer to: http://www.bluesunrise.com/jetspeed-docs/PortletHowTo.htm.

Most of the customizations described below will be performed through the use of Jetspeed's portal customization web interface.

Accessing the Jetspeed Interface

To access the Jetspeed interface

1. Locate SiebelICMConfig.xem(<Install Dir>\jboss-3.0.3_tomcat-4.1.12\SiebelICMconfig).

   NOTE: These changes can be made while the Siebel services are running and will take effect when the file is saved.

2. Change the value of property dashboard.customization to true.

   NOTE: This text must be entered in all lower case.

Creating a New Pane

To create a new pane

1. Log in to Siebel Incentive Compensation Management as a user with an OU administrator role.
2. Click on the dashboard you wish to customize (Administrator, Executive, Manager, or Participant).

   NOTE: There are new portal links in the top right corner of the dashboard's content area.

3. Click the Customize link in the top right corner of the dashboard.
4. To add a new tab to the current dashboard, click Add Pane.

5 Enter the name of the tab pane in the Type field.

NOTE: This will be the name of the hyperlink generated for the new tab.

6 Click Apply.

7 Select the new pane by clicking the pane name at the bottom of the screen.

8 Select the appropriate layout from the Layout drop-down list.

NOTE: Use the "One Column" layout rather than "Single Column", this is a Jetspeed issue.

NOTE: Refer to the Jetspeed user documentation for examples of the other available layouts: http://jakarta.apache.org/jetspeed/site/customizer.html.

9 Click Save and Apply.

Editing an Existing Pane in the Incentive Compensation Management Dashboard

To edit an existing pane

1 Select a pre-existing or newly created tab by clicking the link in the Customize Pane screen.

2 Click Add Portlet.

3 Choose a portlet by name, or to filter by category, use the drop-down category selector. SiebelICM specific portlets will be under a single category: Siebel ICM.

For a list of portlets, see "Portlets" on page 143.

4 Click Apply.

5 Repeat steps 3-4 to add as many portlets as necessary for the layout you have chosen.

6 The new portlet(s) will appear and work properly at this time. Many of the available portlets may be customized using the per-instance portlet customizer that is available for each portlet instance—note that this customizer is only accessible from the customize icon in the title bar of each portlet instance, so if a portlet is registered (see Portlet Registration below) with the title bar disabled, you will not be able to customize its parameters. An example of this is provided below using the "New Web Site" portlet available in the standard Siebel Incentive Compensation Management distribution.

To Add a New Web Site Portlet

To add a new web site portlet

1 First, follow the steps in "Editing an Existing Pane" above, and select the portlet with title "New Web Site". This is a custom Siebel Incentive Compensation Management portlet designed to facilitate the addition of new web sites without having to edit Portlet Registry files by hand.

2 Click on the customization (left-most) icon on the right side of the portlet's title bar, and you will be presented with the per-instance portlet customization screen. Here you can edit any of the portlets parameters:

   a Set the "Title" parameter to whatever you like, but this will usually be the title of the web site, for instance "Yahoo!".

   b Set the "URL" parameter to the web address of the site you wish to display. This must be a fully qualified URL, for example "http://www.yahoo.com".

   c The "open_in_popup" parameter controls whether links within the portlet will spawn a new window, or fill the entire browser screen. The default is to open a new window.

   d The other parameters function exactly as described for "WebPagePortlet": http://jakarta.apache.org/jetspeed/site/portlet_config_WebPagePortlet.html

3 In the "Security ID" select list, choose "Users +V and Admin +C". This will eliminate the icons that appear in the title bar. Note that once you have done this, you can no longer adjust the parameters of the portlet instance through the interface (only manually in the PSML entry). You may, of course, remove this portlet from the tab pane and start over if you make a mistake in configuring the parameters.

4 You should no see your website in your dashboard.

About the New Iframe Portlet

Note that not all web sites will be displayed properly using the "New Web Site" portlet, due to the complex nature of displaying another site's content within a web page—the html retrieved by the portlet is parsed and rewritten as well to remove some elements as that might interfere with the page's operation (JavaScript, applets, etc.), but this also may prevent the site's content from rendering properly. Sometimes changing the instance parameters to not remove certain page elements like JavaScript may help the page render correctly, but if it does not, then any page may be accurately displayed using the "New IFrame Web Site" portlet. This portlet retrieves content in an <iframe> tag, which basically makes the portal pane behave like its own browser window. You do, however, run the risk of adding extra scroll bars when the content will not fit in the portal area. To use this portlet, follow the same steps outlined in "New Web Site" portlet above, but instead of the "URL" parameter, the "source" parameter controls the web site content to retrieve. This portlet accepts the following parameters to control the <iframe> tags that are generated:

- **source** - The target of the <iframe>, where it grabs its content from. Default is "http://0.0.1"
- **width** - The width of the <iframe>, or null to let the browser decide. Default is null.
- **height** - The height of the <iframe>, or null to let the browser decide. Default is null.
- **scrolling** - How to display a scrollbar. Default is "auto", to let the browser decide.
- **frameborder** - Whether or not to display a border around the <iframe>. Default is 1 (yes).

Editing Jetspeed XML(PSML) Files Manually

To manually edit Jetspeed XML (PSML) files

1. Stop your web server.
2. Locate the default.psml file with a text editor.<Install Dir>/jboss-3.0.3_tomcat-4.1.12\server\default\Siebel\Siebel.war\WEB-INF\psml\user\<dashboard_name> \html. where <dashboard_name> is the name of the dashboard you have customized above.

   NOTE: More information about the PSML (Portal Structure Markup Language) format can be found at:
   http://jakarta.apache.org/jetspeed/site/psml.html
3. Locate the portlet entry(s) for the portlet(s) you just added. Perform a "Find" on the portlet name.

   The name will appear within the <title> tag for the portlet entry.

   EXAMPLE: For a new tab pane with a One-Column layout, the entry will look something like this:

```
<portlets id="P-f0ea6458d0-10009">
        <metainfo>
            <title>My Tab Pane</title>
        </metainfo>
        <controller name="OneColumn"/>
        <entry id="P-f0ea656d0e-1000a" parent="Registered Portlet Name">
            <layout position="-1" size="-1">
                <property name="column" value="0"/>
                <property name="row" value="0"/>
            </layout>
        </entry>
</portlets>
```

To disable the customization icons manually, you must add a security reference tag to this XML descriptor as follows:

```
<portlets id="P-f0ea6458d0-10009">
        <metainfo>
            <title>My Tab Pane</title>
        </metainfo>
        <security-ref parent="user-view_admin-all"/>
```

```
<controller name="OneColumn"/>
<entry id="P-f0ea656d0e-1000a" parent="Registered Portlet Name">
        <layout position="-1" size="-1">
                <property name="column" value="0"/>
                <property name="row" value="0"/>
        </layout>
</entry>
</portlets>
```

4 Insert the security reference tag for all the new portlet entries that were created through the Jetspeed interface.

NOTE: This reference tag should be entered just below the </metainfo> closing tag.

Locking the Dashboard Configuration

To lock the dashboard configuration

1 In the install.properties files, change the "dashboard.customization" property to false.

2 Run "setup jboss".

3 Run "setup tomcat".

About Portlet Registration

Each new portlet must be added to the Jetspeed portlet registry. Specific steps to perform this function can be found in the Jetspeed documentation at http://jakarta.apache.org/jetspeed/site/psml.html .

Information is provided below on several different Jetspeed portlets including the Siebel Incentive Compensation Management custom portlets, with general use cases and a high-level description of how each works, as well as information specific to Siebel Incentive Compensation Management's Jetspeed integration. For more information on other portlets available with the Jetspeed distribution as well as configuration information access http://jakarta.apache.org/jetspeed/site/catalog.html.

CAUTION: If his highly recommended you consult the documentation listed above before moving forward through this section.

For information on creating your own custom portlet class, see:

http://www.bluesunrise.com/jetspeed-docs/PortletHowTo.htm

Siebel Incentive Compensation Management's default portlet registry file (SiebelICM-portlets.xreg) can be found in the following location:

<*Install Dir*>\jboss-3.0.3_tomcat-4.1.12\server\default\Siebel\Siebel.war\WEB-INF\conf

You may add portlet registration entries to this file, or create a new file in this directory with your entries. All files with the .xreg extension are loaded by Jetspeed on startup. You may refer to the portlets.xreg, and demo-portlets.xre files for portlet registry examples.

WebPagePortlet

This portlet does exactly what its name implies, which is retrieve HTML content from a given absolute URL and display it within a portlet pane. This should usually be used for data views only, as any forms it contains will not work properly. Any hyperlinks in the retrieved content will be displayed in the full browser window, not within the portal pane. This portlet has myriad options for "massaging" the retrieved HTML content before displaying it (to remove header tags, etc.), which can be disabled via properties in the registry entry.

NOTE: The other portlets described here do not strip HTML header tags (like <html>, <head>, <body>) as does the WebPagePortlet, so care must be taken that they are not part of the returned content (i.e., remove them from your JSP, servlet, etc.). This does not apply to IFramePortlet, as its content is treated as a completely separate web page within an iframe (header tags required).

IFramePortlet

The IFramePortlet is also used to retrieve HTML content from an absolute URL, but it simply generates an appropriately sized iframe with the URL you specify as its "src" attribute. This is useful for creating interactive views, in which the user submits forms or follows navigation links within the portal pane. The downside of this is, of course, that you are using a frame, so you may end up with either unwanted scrollbars or chopped content if the returned content is to large to fit in the area specified for the portlet during customization. Use with care.

JspPortlet

This portlet is useful if you need to create some custom dynamic content of your own to be displayed in the portal, or even if there is just some basic static content you wish to display, like company contact information, that you want by itself (not mixed with other content). Simply create a valid JSP file and place it in this directory:

<web_app_path>\WEB-INF\templates\jsp\portlets

You may import and use any classes you wish in the JSP; all classes available to the Siebel Incentive Compensation Management web application are available to these JSPs as well. Note that although this JSP is invoked though Jetspeed, any paths within it will be resolved relative to the web container's web applications root folder.

ServletInvokerPortlet

You can use this portlet in lieu of the JspPortlet, if you wish, as well as for any other servlet belonging to the Siebel Incentive Compensation Management web application. It accepts a URL relative to the Siebel Incentive Compensation Management web context to specify JSP files, or the relative URL your servlet is mapped to, via a parameter tag in the registry entry you create for it:

```
<parameter name="url" value="/path/to/foo.jsp" hidden="false"
```

The ServletInvokerPortlet grabs a handle to the web container's RequestDispatcher to service the URL you specify (first it prepends it with the web application context path). Keep in mind that another servlet, namely Jetspeed's controller servlet, has already been called before this portlet is invoked, so certain servlets will not work properly, particularly other controller servlets or servlets that directly manipulate the HttpServletResponse. The Struts action servlet falls into both of these categories, meaning this portlet cannot be used for Struts-mapped URLs. To include the output from a Struts-mapped URL in your portlet, use the ServletProxyPortlet below.

ServletProxyPortlet

The portlets discussed to this point are standard Jetspeed portlets included with the distribution. This portlet, however, is part of the Siebel Incentive Compensation Management codebase. It was created to surmount the problem discussed above in the "ServletInvokerPortler" section, namely that Struts-mapped URLs fail when invoked via the ServletInvokerPortlet. The reasons for this are many, but in a nutshell there are "too many generals" vying for control: Jetspeed's controller servlet has already manipulated the HttpServletResponse, so the Struts action servlet throws exceptions almost immediately, because it "thinks" the HttpServletResponse has already been committed.

The ServletProxyPortlet does NOT hand off control to the RequestDispatcher as does ServletPortlet; instead, it opens an internal java.net.URL connection to the specified URL to get the HTML content. For all intents and purposes, this connection is a completely separate internal client; for that reason, the session ID is appended to the URL so that the proxy request will be treated as part of the same HttpSession that the original request belongs to.

ServletProxyPortlet also uses a parameter element in the registry entry to specify the web-context relative URL—note that it uses all uppercase "URL" for the parameter name. A query string may be appended to the URL to pass parameters.

```
<parameter name="URL" value="/path/to/foo.jsp?myParam=false" hidden="false"
```

Overview of Incentive Compensation Management Statements

Siebel ICM statements are based on Jasper Reports, an open-source Java report generation tool. More information and detailed documentation on designing Jasper Reports can be found here:

http://jasperreports.sourceforge.net/

Siebel ICM statement design conforms to the Jasper Report design specification, with the exception of stylesheet integration, described in the section below.

Customizing Statements for Your Installation of Incentive Compensation Management

The process of customizing statements has many steps. They are detailed below:

Configure Statement Security

Statement security is configured by editing the report_groups.txt and reports_acl.txt files specified in the Siebel-config.xml.in file (reporting.report.aclfile and reporting.report.reportgroupfile properties). By default, these files can be found in the ${SiebelICM Install.dir} and are deployed to ${AppSevrerDir}/reports/security. These files are used by the statement portlets to determine user access to statement categories, assign deployed statements to categories, and provide user access to the statements themselves.

After updating the security files you will have to restart the application server before the changes will take effect.

Define Report Categories in report_acl.txt

The report_acl.txt file relates the statement groups defined in report_groups.txt to Operating Unit-specific security roles. Each line in this file maps an Operating Unit Role to a report Group. Note the OU codes, and Role names must match the exact values stored in the transaction database. In addition, the Group codes must match the exact values defined in report_groups.txt. The file is a comma-delimited list with the following column headers:

```
<OUCode>.<Security Role Code>, GroupCode1, GroupCode2, GroupCodeN
```

For Example, these entries map all of the roles for the TIPMSOU and OU_B1 Operating Unites to the Payment report group:

```
TIPMSOU.Participant, Payment

TIPMSOU.Manager, Payment

TIPMSOU.ADMIN ROLE: TIPMSOU, Payment

TIPMSOU.ManagerDashboard, Payment

OU_B1.Participant, Payment

OU_B1.Approver1, Payment

OU_B1.Approver2, Payment

OU_B1.AccountMgr, Payment

OU_B1.SalesRep, Payment
```

Define Report Categories in report_groups.txt

The report_groups.txt file defines categories of statements for display in the report menu. Each line in this file defines a category of statements for a single Operating Unit. The file is a comma delimited list with the following column headers:

```
<OUCode>.<Category Name>, ReportCode1, ReportCode2, ReportCodeN
```

For Example, these entries define one report category (called "Payout") for two OUs, with a single report in each category:

```
OU_B1.Payout, ParticipantPayoutStatement
```

```
OU_B2.Payout, ParticipantPayoutStatement
```

If modifying these parameters in the extract or source directory, run the appropriate script (setup jboss for the distribution, and all for the source) to move the files over.

Report Parameters

The Jasper Reports engine allows for report parameters to be passed in to a report. Parameters are simply object references that define how the report is filled. Siebel statements use integrated parameter handling with the ICM application for online viewing of statements. This integration is a powerful mechanism to enable secure rollout of your online statements as an integrated solution with your ICM application.

Parameters are commonly used as conditions in the data source for your report. Parameters take the following format within the statement design:

```
<parameter name="Foo" class="java.lang.String" isForPrompting="false" >
    <parameterDescription>
        <![CDATA[
            StatementParameterHandler
        ]]>
    </parameterDescription>
    <defaultValueExpression>
        <![CDATA[
            "Bar"
        ]]>
    </defaultValueExpression>
</parameter>
```

Java Classes for Parameters

Each parameter is assigned a java class to define the type. The parameter type will depend on the use of the parameter. If using the parameter as a report variable its type must be identical to the variable type. If used as a textFieldExpression, it must also be the same as the text field. See the Jasper Reports documentation for more details on variables and text fields.

The following parameters all have special purposes for the ICM Statements module:

- IsForPrompting
- parameterDescription
- defaultValueExpression

Integrated Parameter Handling

Integrated parameter handling allows you to deploy and secure ICM Statements for online delivery through the ICM Portal. Understanding how the Statements module applies parameter handling is essential to delivering quality reports.

By default, all Statements are passed "system" parameters. System parameters are not required in the Statement, but are available to the Statement developer:

- sys_OperatingUnitCode (java.lang.String) - the operating unit code from the session of the user accessing the statement
- sys_participant_user_uuid (java.lang.String) - the participant code from the session of the user accessing the statement. This parameter integration is designed for use in integrated data security conditions. See section below for more information on data security rules and use within ICM statements.
- sys_EnterpriseUnitCode (java.lang.String) -- the enterprise unit code from the session of the user accessing the statement
- sys_PeriodNumber (java.lang.Integer) - the current working period number from the session of the user accessing the statement.

System parameters must be declared in the statement design, for example:

```
<parameter name="sys_OperatingUnitCode" class="java.lang.String" isForPrompting="false"/>
```

After declared, these system parameters are available in the rest of the report using the $P{parameter_name} convention. See the Jasper Reports documentation for more detail on using parameters in report designs.

Parameter Handlers

The Siebel ICM Statements module can present statement parameters defined as prompts to the end user accessing the statement online. The prompt behavior is defined in parameter handlers that are delivered with the Statements module. To use a parameter handler to within a statement, the <parameterDescription> element is declared with a CDATA section that contains the parameter handler to be applied. For example:

```
<parameter name="ParticipantCode" class="java.lang.String" isForPrompting="true" >
    <parameterDescription>
        <![CDATA[
            PromptHandlerClass:ParticipantCodePromptHandlerImpl
        ]]>
    </parameterDescription>
</parameter>
```

Delivered Parameter Handler Description and Behavior

PeriodPromptHandlerImpl

Creates a drop-down list of all the periods for the current Operating Unit.

preparePrompt sets the following prompt field values:

Label	Period
DisplayType	Select
DisplayWithReport	True
Valid Values	Collection with one element for each period. Each element is a pair of the form: (period end date, absolute period number)

validate - no validation

StartPeriodPromptHandlerImpl

Identical to PeriodPromptHandlerImpl, except the label is "Start Period End Date:".

EndPeriodPromptHandlerImpl

Identical to PeriodPromptHandlerImpl, except the label is "End Period End Date:".

ComparisonPeriodPromptHandlerImpl

Creates a drop-down list of all the periods for a given calendar year, for use in the comparison portlet. Uses the 'CalendarYearUuid1' parameter of the comparison portlet, and returns a collection with one element for each period in that calendar year.

preparePrompt sets the following prompt field values:

Label	Period
DisplayType	Select
DisplayWithReport	True
Valid Values	Collection with one element for each period. Each element is a pair of the form: (period number, period number)

validate - no validation

ComparisonStartPeriodPromptHandlerImpl

Identical to ComparisonPeriodPromptHandlerImpl, except the label is "Start Period End Date:".

ComparisonEndPeriodPromptHandlerImpl

Identical to ComparisonPeriodPromptHandlerImpl, except the label is "End Period End Date:".

MeasurePeriodPromptHandlerImpl

Creates a drop-down list of all the measures active in the current period.

preparePrompt sets the following prompt field values:

Label	Period
DisplayType	Select
DisplayWithReport	False
Valid Values	Collection with one element for each measure. Each element is a pair of the form: (measure code, measure uuid)

validate - validates that the selected measure exists. This validation is mainly redundant based on the existing preparePrompt, but provides an extra level of error trapping, and allows this handler to be extended to other display types in future.

ParticipantCodePromptHandlerImpl

If the user is a participant user, returns the participant code of that user as a hidden field. If the current user is not a participant, returns '%'.

preparePrompt sets the following prompt field values:

Label	null
DisplayType	Hidden
DisplayWithReport	False
ValidValues	null

validate - no validation

ParticipantCodeUsingTextboxPromptHandlerImpl

If the user is a participant user, returns the participant code of that user as a hidden field. If the current user is not a participant, displays a textbox for participant code selection.

preparePrompt sets the following prompt field values (for participant/non-participant):

Label	null/Participant Code:
DisplayType	Hidden/Textbox
DisplayWithReport	False
Valid Values	null

validate - no validation is done at this time. Participant codes entered in the textbox will be validated when the report is filled, based on security information in the Analytics DB.

Data Security

Data access to various objects is controlled based on the data that resides in the DW_USER_SECURITY_MAP table. The data in this table defines the set of participants that each user should may access, and the table is populated as part of the Update Analytics service based on transactional system setup. A simple example:

Assume there is a user (admin_user_1) and four participants (admin_user_1, P1, P2 and P3) in the system. If admin_user_1 should have the ability to see data that is associated to all of the participants, the DW_USER_SECURITY_MAP table will have four corresponding rows of data:

```
user_uuid       user_security_map_code

--------------  -------------------------

admin_user_1    admin_user_1

admin_user_1    P1

admin_user_1    P2

admin_user_1    P3
```

If the admin user should only be able to access the date related to admin_user_1 and P3 then there should only be two rows of data that correspond to admin_user_1.

```
user_uuid       user_security_map_code

--------------  -------------------------

admin_user_1    admin_user_1

admin_user_1    P3
```

The actual schema of the DW_USER_SECURITY_MAP (Oracle example):

```
user_uuid                VARCHAR2(36) NOT NULL,
user_security_uuid       VARCHAR2(36) NOT NULL,
user_security_type_flag  VARCHAR2(1) NOT NULL,
user_security_code       VARCHAR2(30) NOT NULL,
user_security_name       VARCHAR2(255) NULL,
unit_code                VARCHAR2(30) NOT NULL,
user_name                VARCHAR2(50) NOT NULL,
publish_date             DATE NOT NULL,
last_update_date         DATE NOT NULL
```

Associated Column Definitions

Column	Description
user_uuid	Internal unique identifier that identifies the Siebel user that this row of data relates to. This user will have the ability to access data associated to the participant that is identified by the user_security_uuid. This will be a USER_CE.user_uuid.
user_security_uuid	Internal unique identifier that identifies the Siebel participant that this user has access to. This will be a PARTICIPANT.participant_uuid.
user_security_type_flag	Identifies the mechanism of security hierarchy that was used (this will be described further in the next section that describes how the data is populated in this table). Current values are 'P' (participant based), 'O' (organization based) and 'T' (territory based).
user_security_code	If the user_security_type_flag is 'P', then this is the participant_code of the participant identified by the user_security_uuid. If the user_security_type_flag is 'O', then this is the organization_code of the organization identified by the user_security_uuid. If the user_security_type_flag is 'T', then this is the territory_code of the territory identified by the user_security_uuid.
user_security_name	This is the participant_name, organization_name or territory_name of the entity identified by the user_security_uuid.
unit_code	This is the operating_unit_code that this data is related to
user_name	This is the user_name of the user identified by the user_uuid.
publish_date	Date of when the data was published.

Rules for DW_USER_SECURITY_MAP Population

OUs, EUs, and Special-Privileg Users

Operating unit administrators, enterprise unit administrators, and any user who has one of the following role privileges:

- Enterprise super user (OU admin)
- Executive dashboard user
- Admin dashboard user

These users will be given security access for each participant, organization and territory in the operating unit that is being published. The reason the additional access to organizations and territories is provided is because credits, cumulated credits, goals and cumulated goals can be associated to participants, organizations or territories.

Each data row will specify the actual mapping type used "P' for participants, 'O' for organizations and 'T' for territories in the user_security_type_flag field.

User Access

Each User should be able to access his/her own related data. So there is a mapping entry for each user and their associated participant code. This data will have a user_security_type_flag = 'P'.

Managers of Organizations

Organization Managers will be able to access all the participants in the organizations beneath them and of the one that they manage and see each employee's active job (currently there is only 1 active job allowed for the employee which is defined by the DW_EMPLOYEE_JOB.active_flag=1). This will have a user_security_type_flag = 'P'.

These users will also be able to access all the of the organization related data to the organizations that are below them in the hierarchy, including the one that they manage. These entries will have a user_security_type_flag = 'O'.

Managers of Channel Partners

Each user who is a channel manager of a channel partner (as defined by the DW_CHANNEL_PARTNER.channel_mgr_participant_uuid) will be able to access data that is related to that channel partner. This will have a user_security_type_flag = 'P'.

Managers of Customers

Each user who is an account manager of a customer (as defined by the DW_CUSTOMER.account_mgr_participant_uuid) will be able to access data that is related to that customer. This will have a user_security_type_flag = 'P'.

Managers in Territories

They will be able to look at all the of the territories related data to the territories that are below them in the hierarchy, including the one that they manage. These entries will have a user_security_type_flag = 'T'.

Statement Stylesheets

Siebel has extended the Jasper Architecture to integrate css support for the presentation of your statements through the ICM application. By default, deployed statements will inherit the styles defined in your ICM application in the Transaction and Modeling .css files.

To use a .css class, you include the cssClass element in the reportElement tag. For example, to use the css class "reportDetail1Right", you would include it as follows:

```
<reportElement isRemoveLineWhenBlank="true" isPrintWhenDetailOverflows="false"
cSSClass="reportDetail1Right" x="600" y="0" width="200" height="15" />
```

NOTE: css class styles are only applied to HTML views of a Statement. PDF statement views do not inherit the css styles. PDF formats must be applied to report elements as per standard Jasper Reports design specifications.

Default Dashboard Configuration

The following table describes the default configuration of the five dashboard types provided standard with Siebel ICM. These types are: Participant, Manager, Executive, Administrator, and Planning.

Participant Dashboard

Panes	Layout	Portlets	Implements	Comments
Participant Home	50/50			
		Simple Statement Portlet	ParticipantCumulatedPerformance	Display current to-date performance based on cumulated credits/goals.
		iFrames	www.siebel.com	Example to show an external website.
Payouts and Disputes	1 column			
		Participant Inbox	Workflow Inbox	Inbox with dispute status.
		Participant Payouts Tab	Payout Review	Review of payout information for current period.
Statements	1 column			
		Participant Plans Tab	Eligible Plans Review	
Eligible Plans	1 column			
		Participant Plans Tab	Eligible Plans Review	Shows listing of eligible plans. Allows user to navigate to plan definition.

Manager Dashboard

Panes	Layout	Portlets	Implements	Comments
Manager Home	50/50			
		Participant Plans Tab	Eligible Plans Review	Shows listing of eligible plans. Allows user to navigate to plan definition.
		Simple Statement Portlet	ParticipantCumulativePerformance	Display current to-date performance based on cumulated credits/goals.
Organization Performance	Single Row			
		Simple Statement Portlet	ParticipantCreditRankingByOrganization	Shows ranking of participants for the manager organization as per the DW security rules.
Manager Statements	1 column			
		Statements	Categories Statement View	See Statements specification for details on navigation.

Executive Dashboard

Panes	Layout	Portlets	Implements	Comments
Executive Home	50/50			
		Participant Plans Tab	Eligible Plans Review	Shows listing of eligible plans. Allows user to navigate to plan definition.
		Simple Statement Portlet	CommissionDetailsByCustomer	Display commissions paid per customer for current period.

Panes	Layout	Portlets	Implements	Comments
Organization Performance	Single Row			
		Simple Statement Portlet	ParticipantCreditRanking ByOrganization	Shows ranking of participants for the manager organization as per the DW security rules.
Executive Statements	1 column			
		Statements	Categories Statement View	See Statements specification for details on navigation.

Administrator Dashboard

Panes	Layout	Portlets	Implements	Comments
Status	1 column			
		Administrator Status	Status	Shows review of service executions.
		Simple Statement Portlet	CommissionDetailsByCus tomer	Display commissions paid per customer for current period.
Administrator Statements	1 column			
		Statements	Categories, Statement View	See Statements specification for details on navigation.

Planning Dashboard

Panes	Layout	Portlets	Implements	Comments
Home	50/50			
		Goal Attainment	GoalAttainmentAllPeriods	Shows review of service executions.

Panes	Layout	Portlets	Implements	Comments
		Payout Distribution	ParticipantPayoutDistribution	Display commissions paid per customer for a current period.
Comparison Portlet	1 column			
		Comparison	Comparison (Modeling)	See Statements specification for details on navigation.

Delivered Reports

The following tables describe standard reports provided with Siebel ICM.

GoalAttainmentAllPeriods

Default Category. Performance

Data Security Implementation. None

Portlet(s) Supported for Deployment. Goal Attainment (Modeling)

Target Dashboard. Planning

Statement Description. Sum of a credit reading and a goal reading for a period range. This report is designed to simply show the credit sum and goal sum for a variable period range, if a cumulation is not used. It is a period range report, using a single credit and goal reading that are fixed parameters in the report design.

ParticipantCreditRankingByOrganization

Default Category. Performance

Data Security Implementation. User

Portlet(s) Supported for Deployment. Simple, Statement.

Target Dashboard. Executive, Manager, (Organization Performance Pane)

Statement Description. Sum of a credit reading per participant for a given period, with the report grouped by Organization Code (selected from employee job). This is a single period, current period report designed to be run in the current period. This is looking at current org assignments for the participant. The credit reading and measure code are hard-coded parameters into the report.

ParticipantCreditRankingByRegion

Default Category. Performance

Data Security Implementation. User

Portlet(s) Supported for Deployment. Simple, Statement

Target Dashboard. Executive, Manager (Organization Performance Pane)

Statement Description. Sum of a credit reading per participant for a given period, with the report grouped by Territory. This is a single period, current period report designed to be run in the current period. The credit reading and measure code are hard-coded parameters into the report. Functionally identical to the CreditRankingByOrganization, but for use with Territory implementations.

ParticipantCumulatedPerformance

Default Category. Performance

Data Security Implementation. User

Portlet(s) Supported for Deployment. Simple, Statement

Target Dashboard. Participant (Participant Home, 50/50)

Statement Description. Narrow report showing the cumulated goal, credit, and percentage (over or under) goal for a single user, or a user list based on user security access. This statement is a narrow design to fit easily on a 50/50 participant portlet or administrator portlet. This is a single period, current period report designed to be run in the current period to give "to-date" achievement information. The credit reading, goal reading and measure code are hard-coded parameters into the report.

ParticipantPerformanceCurrentPeriod

Default Category. Performance

Data Security Implementation. None

Portlet(s) Supported for Deployment. Comparison, Statement

Target Dashboard. Planning

Statement Description. For a given plan, measure the participant performance as a percentage of goal/credit for the current period. Similar to the cumulated performance report, but for the current period only for a given plan. The credit and goal readings are parameters, hard-coded into the report design. The measure uses the measure prompt handler. The plan code can be manipulated through the modeling portlet.

ParticipantTransactionsByPeriod

Default Category. Transaction

Data Security Implementation. User (ParticicipantCodePromptHandler)

Portlet(s) Supported for Deployment. Statement

Target Dashboard. All

Statement Description. This report is a period range report that enables a transaction view secured by the DW security map for given users associated with the sales transaction header or line. It is a listing of transactions grouped by participant, showing the product and customer for each line, as well as a single line reading. This line reading is a hard-coded parameter in the report. The sum of the line reading is specified as a group footer (sum per participant for the period range).

PayoutDistributionByPlan

Default Category. Transaction

Data Security Implementation. None

Portlet(s) Supported for Deployment. Payout Distribution (Modeling)

Target Dashboard. Planning

Statement Description. Modeling dashboard report -- breakdown of participants that fall into a payout distribution, per plan. The distribution is 5 categories, representing 20% increments between the highest and lowest bounds of the payout for the current period. This is a current period, single period report that is optimally used with the modeling portlet designed to change the plan dynamically.

PerformanceRankingByPeriodMeasure

Default Category. Transaction

Data Security Implementation. None

Portlet(s) Supported for Deployment. Statement

Target Dashboard. All

Statement Description. Ranking of participants earning credits in the current period for a given measure. This is a current period, single period report that uses the measure prompt handler for dynamic measure navigation from within the statements portlet.

PlanParticipantPayout

Default Category. Payout

Data Security Implementation. User

Portlet(s) Supported for Deployment. Statement

Target Dashboard. All

Statement Description. For a period range, this report will sum all calculated results, with the formula code used and the incentive type code; grouped by plan and period number. This is a very useful example for plans with important calculated results, as well as details on how incentive types were calculated per period per plan. This is a period range report, although the output should be grouped and sum (sum of calculated results) per individual period.

RevenuePayoutAllPeriods

Default Category. Transaction

Data Security Implementation. None

Portlet(s) Supported for Deployment. Comparision, Statement

Target Dashboard. Admin Access Only

Statement Description. This report was built to provide an easy report to be used in the comparison portlet. It sums the revenue (as defined by the sum of a sales transaction reading given a specific event, grouped by period) for a period and compares that revenue against the total payouts for that period. A column with the percentage of "revenue" that is paid for all participants. In a scenario, this would be used to see how plan changes impact payouts.

TransactionSumForEventByProduct

Default Category. Transaction

Data Security Implementation. User

Portlet(s) Supported for Deployment. Comparision, Statement

Target Dashboard. All

Statement Description. This report displays a sum of a transaction field grouped by product for a given period range. This is a period range report, and should use the start and end period prompt handlers to allow for navigation through the statements portlet. Notice the user data security requirement -- this is an excellent report for participants. It also would serve as a good example for an identical report grouped by customers.

CommissionDetailsByCustomer

Default. Payout

CommissionStatement

Default Category. Payout

Data Security Implementation. User

Portlet(s) Supported for Deployment.

Target Dashboard.

Statement Description. See the Commission Statement specification for this new statement. Note that this is a single period statement, and should use the ParticipantCode prompt handler to prompt the user for a specific Participant Code if the user is not a participant. There is no '%' functionality with this report -- the commission statement is always viewed for a single participant, for a single period.

CustomerSalesVsCommission

Default Category. Payout

Data Security Implementation. User

Portlet(s) Supported for Deployment.

Target Dashboard.

Statement Description. This report should sum all payouts and "revenue" tied to transactions, grouped by the customer on those transactions. The commission paid is driven by the formula final result for a give period range (this is a period range report), and the transactions sum should reflect that period range as well (based on transactions that led to credits that led to the payout. This is a participant report, and should serve as an example of measuring the worth (in terms of payout per $ of customer sales) over a given period range.

Notes

- Period range reports use the start period and end period prompt handlers.
- Single Period reports use the period prompt handler.
- Single Period, Current Period reports use the sys_PeriodNumber input parameter in the query.
- Single Period, Current Period reports use the sys_PeriodNumber input parameter in the query.
- Data security implementations are user (as per the DW security map security rules) and none (no security implemented, always return all rows).
- Data security implementations are user with the participant code prompt handler will pass the participant ID to the query, or a % if the user is a non-participant user.

Portlets

This section describes the different types of portlets used by the standard Siebel ICM dashboards.

Table 1. Portlets

Delivered Portlet	Dashboards to Implement	Functionality Comments
Comparision	Planning	Comparison portlet. See comparison document. Should only be implemented in planning in a single column pane
Statements	All	Statement category display, integrated prompt handling, and display. Navigation determined by prompt handlers (see prompt handler information). Should display and re-execute report when different prompts selected. Should always be implemented in its own, single column pane. Export to PDF available.

Table 1. Portlets

Delivered Portlet	Dashboards to Implement	Functionality Comments
Goal Attainment	Planning	Should implement the GoalAttainment.jsp in a 50/50 pane (2 columns) on the planning dashboard.
Payout Distribution	Planning	Should implement the PayoutDistribution.jsp in a 50/50 pane (2 columns) on the planning dashboard.
Simple Jasper	All	Executes a report passing only the default parameters (see prompt handling document for details). Can be used to display any report requiring only the default prompts passed. Report page and column width will determine where to place it on the dashboard. Export to PDF available.

21 Log Files

Logs

The ability to track errors and processes happening in the application is a powerful tool for troubleshooting your Siebel ICM implementation. Several different types of log files are provided with the application and the settings within the logs can be adjusted as required. There are five different settings available for the log files; Debug, Info, Warn, Error, and Fatal. The more detailed settings (Debug, Info, and Warn) can create very large files and can impact the overall performance of the software. Console logging has a much greater impact on performance than File logging. It is recommended to turn off all Console logging when in a production environment. These settings should be used only at the direction of Siebel Professional Services.

For more information see the *Jakarta Log4J Short Manual.*

Application Log

The Application Log tracks normal application use. Tasks tracked include, logging into the application, selecting entities from the main menus, creating new entities etc. This log file can be found in the following directory:

<*Install Drive*>\jboss-3.0.3_tomcat-4.1.12\bin

Appenders and Categories

Each log file configuration consists of an appender (where the output should go), and a category (what types of messages are logged). Only advanced users should change the appenders; all changes to logging levels are made in the category declaration.

To change the level of logging in this file open the log4j.xml file with a standard text editor from the following location:

<*Install Drive*>\jboss-3.0.3_tomcat-4.1.12\server\default\conf

Appenders can be configured to change the name/location of files that are being logged to, the message formats, file size maximums, and maximum number of files to keep for historical purposes.

The Console Appender

The Console appender will send logged output to the application server startup window, if the server is running in standalone, non-service mode.

```
<appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">

   <param name="Target" value="System.out"/>
```

```
    <layout class="org.apache.log4j.PatternLayout">
    <!-- The default pattern: Date Priority [Category] Message\n -->
    <param name="ConversionPattern" value="%d{ABSOLUTE} [%c{1}] %m%n"/>
    </layout>
</appender>
```

The Advanced Rolling File Appender

The advanced rolling file appender logs to a file located in <*Install Drive*>\jboss-3.0.3_tomcat-4.1.12\bin. This appender exists to separate infrastructure debugging/logging information from application specific debugging/logging information.

```
<appender name="DEVRFA" class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="Siebel_Advanced_Log.html" />
    <!-- append to log or restart it? -->
    <param name="Append" value="false" />
    <param name="MaxFileSize" value="4096KB" />
    <param name="MaxBackupIndex" value="1" />
    <!-- this layout will create html -->
    <layout class="org.apache.log4j.HTMLLayout" >
       <param name="LocationInfo" value="false" />
    </layout>
</appender>
```

The Application Rolling File Appender

The application rolling file appender exists to log application level, not application server, debugging/logging information to a file that will be split into individual logs of size 4096 KB.

```
<appender name="RFA" class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="Siebel_Application_Log.html" />
    <!-- append to log or restart it? -->
    <param name="Append" value="false" />
    <param name="MaxFileSize" value="4096KB" />
    <param name="MaxBackupIndex" value="1" />
    <!-- this layout will create html -->
```

```
    <layout class="org.apache.log4j.HTMLLayout" >
        <param name="LocationInfo" value="false" />
    </layout>
</appender>
```

The Application Server Rolling File Appender

The application server rolling file appender exists to send application server (j2ee) logging messages to a server.log file. This can be useful for diagnosing possible issues with the application server infrastructure.

```
<appender name="APPSERVER"class="org.jboss.logging.appender.DailyRollingFileAppender">
    <param name="File" value="${jboss.server.home.dir}/log/server.log"/>
    <param name="Append" value="false"/>
    <!-- Rollover at midnight each day -->
    <param name="DatePattern" value="'.'yyyy-MM-dd"/>
    <!-- Rollover at the top of each hour
    <appender-ref ref="CONSOLE"/>
    <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>
    -->
    <layout class="org.apache.log4j.PatternLayout">
        <!-- The default pattern: Date Priority [Category] Message\n -->
        <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
        <!-- The full pattern: Date MS Priority [Category] (Thread:NDC) Message\n
        <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%t:%x) %m%n"/>
        -->
    </layout>
</appender>
```

The Audit Log Appender

- The Audit log appender exists to send auditing information to a file for manual inspection. Audit events exist for:
- User creation
- Role creation

- Role update
- User update
- Payout adjustments

```
<appender name="AUDIT" class="org.apache.log4j.RollingFileAppender">
   <!-- pick your output file name -->
   <param name="File" value="audit.txt" />
   <!-- append to log or restart it? -->
   <param name="Append" value="false" />
   <param name="MaxFileSize" value="4096KB" />
   <!-- pick the number of backups to be kept, max value is MAXINT-->
   <param name="MaxBackupIndex" value="5" />
   <layout class="org.apache.log4j.PatternLayout" >
      <param name="ConversionPattern" value="[%-5p] [%d{yyyy-MM-dd HH:mm:ss SSS}]
      [%c{1}] %m%n"/>
   </layout>
</appender>
```

In order to change the level of logging for a given category, you must change the category definitions. Each category definition can consist of the following:

Name. The name of the category. Do not change this as internally Siebel IC sends log messages to this name.

Priority. The threshold that must be met or exceeded for the log message to be sent to the appender. Appropriate priorities are (in order): DEBUG, INFO, WARN, ERROR and FATAL.

Appender-ref. The appender you want to send the logged message to. You can specify multiple appenders for a category.

Examples

Messages logged that refer to the dashboard will go to the com.siebel.dashboards category. They are logged to the Rolling File Appender, and have no filtering of log levels.

```
<category name="com.siebel.dashboards">
   <appender-ref ref="RFA"/>
</category>
```

Messages logged to the Audit category will go to the Audit appender, and only messages of level INFO or above will be logged.

```
<category name="Audit">
    <priority value="info" />
    <appender-ref ref="AUDIT" />
</category>
```

System level messages will go to the Rolling File appender and only messages of level INFO or above will be logged.

```
<category name="SiebelICMSystemMessage">
    <priority value="info" />
    <appender-ref ref="RFA"/>
</category>
```

Developer (Advanced) log messages will go to the Advanced Rolling File Appender and all messages of level DEBUG or higher will be logged.

```
<category name="developer">
    <priority value="debug" />
    <appender-ref ref="DEVRFA" />
</category>
```

General debugging messages for the Siebel IC platform will go to this category. Messages of DEBUG or higher will be logged.

```
<category name="com.motiva">
    <priority value="DEBUG"/>
    <appender-ref ref="RFA"/>
</category>
```

The Siebel IC framework category consists of messages for base services; generally this does not need to be configured. If problems with JNDI, or CONFIGURATION occur, the priority can be lowered to DEBUG to get more messages.

```
<category name="FrameWork">
    <priority value="fatal" />
    <appender-ref ref="RFA"/>
</category>
```

Persistence based log messages are logged to the org.jboss.jbo.castor category. If errors are related to persistence, change the priority value to DEBUG to get more information.

```
<category name="org.jboss.jdo.castor">
    <priority value="ERROR"/>
    <appender-ref ref="APPSERVER"/>
</category>
```

Application server messages generated by the JBOSS application server will be logged to the Application Server Log Appender, and only messages of INFO or higher will be logged.

```
<category name="org.jboss">
    <priority value="INFO"/>
    <appender-ref ref="APPSERVER"/>
</category>
```

Apache log messages go to the org.apache category. If problems are encountered with Tomcat, or any other bundled Apache software, change the priority to DEBUG to get more information.

```
<category name="org.apache">
    <priority value="ERROR"/>
    <appender-ref ref="CONSOLE"/>
</category>
```

Boot Log

The Boot Log contains information about the startup sequence of Jboss/Tomcat during application startup. This log file can be found in the following directory:

<*Install Drive*>\jboss-3.0.3_tomcat-4.1.12\server\default\log

The settings for this log file cannot be adjusted.

LocalHost Access Log

The LocalHost Access Log contains information about the activity of the webserver (Tomcat). This log file can be found in the following directory:

<*Install Drive*>\jboss-3.0.3_tomcat-4.1.12\server\default\log

This log file name is appended with the current server date as follows:

localhost_access2003-04-21.log

The settings for this log file cannot be adjusted.

A Context Attributes Reference

The following context attribute script is part of standard Siebel Incentive Compensation Management installations. The explanation of the script will help guide you in generating your own customized scripts for additional context attributes you create. Note that all other standard context attribute scripts in Siebel Incentive Compensation Management follow the same pattern as the one outlined below.

Customer Code

The Customer Code context attribute can be used in credit rules to get the customer code from a transaction header. This makes the customer's code available for generated credit records. The script is as follows:

```
// CaCustomerCode.script

//

// Context Attribute: Customer Code of Sales Transaction
```

These lines are each preceded by "//" which indicates that they are comment lines, and are ignored by the system. These comment lines tell you what the script's name is and what data it will fetch for credit rules.

```
importClass(Packages.com.againtech.framework.logging.LogService);

importClass(Packages.com.againtech.framework.jndi.JNDIService);

var CAT = "com.againtech.install.script.CaCustomerCode.script";
```

This script block is a standard part of all pre-packaged scripts in Siebel Incentive Compensation Management. The first two lines call up standard services that should be included in every script. The next line creates the variable CAT and assigns the script name and location to that variable.

```
// Get Customer object

var customer = SYS_EVENT.getLine().getItem().getCustomer();
```

This line declares a new variable, "customer", and assigns to it the value found in the Customer field on a transaction header. Note that this line is executed in the context of a line and an event on that line because these are always known during crediting. Even though the customer code is found on the transaction header, all transaction lines are considered to inherit the data from the header.

```
if (customer == null)

{

   LogService.info(CAT,"Can't find Customer object in CaCustomerCode.");
```

```
      return;
   }
```

This block covers the possibility that there is no customer listed on the transaction header. If this is the case, the system will send a message to the logging service that the customer code couldn't be found, and the script will return no value for the customer code. This means that any credit rule distribution that attempts to write the customer code to a credit record will simply leave that field blank. This will only cause problems if that field on the credit record is a required field; otherwise the system will not run into any other errors during the credit distribution process.

```
   // get customer code

   return customer.getCode();
```

The final line of the script returns the customer code from the transaction header. This is the value that the system ultimately will write to a credit record that references this context attribute.

B Condition Templates Reference

The following condition templates are found in every standard Siebel Incentive Compensation Management installation. The explanations of the JavaScript codes will help guide you in constructing your own customized scripts and templates.

Line Product Is Specified Product

This condition template is designed to get the value in the Product Code field on a transaction header and compare it to a value that is specified by the user. The following JavaScript code handles this function.

```
// AssrCreditProdIsProd.script
//

// Crediting Assessor Template: Is Transaction Line Product equal to Specified
product

//  Parameters:

//    - productUUID
```

These lines are each preceded by "//" which indicates that they are comment lines, and are ignored by the system. These comment lines tell you what the script's name is, its function, and what parameters will be used by the script.

```
importClass(Packages.com.againtech.framework.logging.LogService);

importClass(Packages.com.againtech.framework.jndi.JNDIService);

var CAT="com.againtech.install.script.AssrCreditProdIsProd.script";

LogService.info(CAT,"Credit Assessor Template: AssrCreditProdIsProd");
```

This script block is a standard part of all pre-packaged scripts in Siebel Incentive Compensation Management. The first two lines call up standard services that should be included in every script. The next line creates the variable CAT and assigns the script name and location to that variable. The last line writes information to the logging service, so that the Create Credits service can note which condition template was used during the process.

```
// Execution Context

LogService.info(CAT,"HASHMAP->" + HASHMAP.toString());
```

This line writes additional information to the logging service. The term HASHMAP refers to the mapping between entity codes that users see in Siebel Incentive Compensation Management and the unique ID codes (the UUIDs) that Siebel Incentive Compensation Management automatically assigns to those entity codes. These "hash" codes are normally hidden from users and cannot normally be accessed or referenced directly. In this case, the script gets the "hash" code for the current instance of the Create Credits process and writes it to the log file for future reference.

```
// Input Parameters

var productUUID = PARAMS.getChild("entity_0").getText();

LogService.info(CAT,"productUUID->" + productUUID);
```

The second line of this block creates the variable "productUUID" and assigns to it the text value that the user entered when the condition template was used in a credit rule. This value will change from one use of the condition template to another. The third line logs this value and labels it for ease of reference.

```
// Get Product from Line corresponding to Event

var lineProduct = SYS_EVENT.getLine().getProduct();
```

This line creates a new variable, "lineProduct," and gets the product code from the transaction line. Note that the product code is retrieved in the context of the line and the line's event. These items do not have to be referenced separately elsewhere in the code because Siebel Incentive Compensation Management automatically knows, when processing transactions through credit rules, what line it is processing and which event on that line it is processing.

It is also worth noting that although the product code is retrieved in the context of a line, the product code is actually recorded on a transaction's header. This is permissible because transaction header data (except for participants) is inherited by the transaction's lines.

```
if ( lineProduct == null )

{

   LogService.info(CAT,"No Product defined for Transaction Line in
AssrCreditProdIsProd.script.");

   return(false); }
```

This condition block deals with cases when no product is referenced on a transaction. The system logs a message in the log file to note that the crediting process failed because no product was found on the transaction, and also returns a value of *false* for the script. The remaining conditional blocks are not executed if this happens.

```
if ( !(lineProduct.getUUID().equals(productUUID)) )

{

  LogService.info(CAT,"Transaction Line Product does not match Product for Rule in
AssrCreditProdIsProd.script.");

   return(false); }
```

This condition block catches all the transaction lines for which the transaction's product code does not match the product code set up for the credit rule. The system logs a message that the crediting process failed for this transaction because it did not meet this condition, and returns a value of *false* for the script.

```
return(true);
```

The final line of the script returns a value of *true* for the script. This necessarily means that the system found a product code on the transaction and that product code met the requirements of the condition. If this template were the only one used in a credit rule, then the transaction line would meet all the rule's conditions and the system would proceed to create distributions for the transaction line.

Line Field (String) <operator> <value>

This template is designed to get data from a custom field on a transaction line, then compare that value to a value that the user has specified for the credit rule. The comparison is accomplished with a conditional operator that the user also chooses. As an example, the user might want the credit rule to check the Sales Amount on a transaction line to see if it exceeds a certain value, say 30,000. The template as used in the credit rule would look like:

Line Field SalesAmount is greater than 30000

The script that goes with this template is as follows:

```
// AssrCreditLineReadingStringCompare.script

//

// Crediting Assessor Template: Compare LineReading string value using given
operator to given value

//  - Parameters:

//      - readingTypeUUID - the reading type to fetch from the Line

//     - operator       - the operator to use (String ops: EQUAL, NOT_EQUAL, LIKE,
NOT_LIKE)

//      - stringValue     - the value to compare the line reading to

//

importClass(Packages.com.againtech.framework.logging.LogService);

importClass(Packages.com.againtech.framework.jndi.JNDIService);

importClass(Packages.com.againtech.ce.util.script.Evaluator);


var CAT = "com.againtech.ce.install.script.AssrCreditLineReadingCompare.script";

LogService.info(CAT,"Assessor Template: AssrCreditLineReadingCompare");
```

These opening lines of the script are very similar to the opening lines of the previous script. Note that an additional tool, the script "Evaluator", is also called. This utility is itself another JavaScript that automatically handles the comparison of one value to another, according to various comparison operators. It will be referenced again near the end of the script (see below).

```
// Execution Context

//

LogService.info(CAT,"HASHMAP->" + HASHMAP.toString());


// input parameters

var readingTypeUUID = PARAMS.getChild('entity_0').getText();

var operator = PARAMS.getChild('operator_0').getText();

var stringValue = PARAMS.getChild('input_0').getText();

// input_0 type is string


LogService.info(CAT,"readingTypeUUID->" + readingTypeUUID);

LogService.info(CAT, "operator->" + operator);

LogService.info(CAT, "stringValue->" + stringValue);
```

This block of script declares three variables, each corresponding to one of the three input values that the user must choose when setting up the template within a credit rule. Note that these lines are normally automatically generated when you set up the initial parameters of the condition template, but the system will assign generic names to these variables. The names used in this script (readingTypeUUID, operator, stringValue) were manually modified so that these variables would be more easily identifiable elsewhere in the script and their function more readily apparent to anyone else that wants to reference the script.

The last three lines of this block log the values that the user chose and labels them so that anyone reading the log report can easily identify the parameters that were chosen.

```
// look up the reading type

var readingTypeServicerHome =
JNDIService.findHome(Packages.com.againtech.ce.readingtype.ReadingTypeServicerHome
.JNDI_NAME);

var readingTypeServicer = readingTypeServicerHome.create();

var readingType = readingTypeServicer.get(readingTypeUUID);

if (readingType == null)

{
```

```
  LogService.error(CAT, "ReadingType not found for readingTypeUUID->" +
readingTypeUUID);

  return(false);}
```

This block of script makes sure that the custom field that the user referenced when setting up the credit rule is a custom field that actually exists in the system. The first half of the block is actually a single function that has been broken into three variable declarations to make the script easier to understand. It could have been written as a single variable declaration:

```
Var
readingType=JNDIService.findHome(Packages.com.againtech.ce.readingtype.ReadingType
ServicerHome.JNDI_NAME).create().get(readingTypeUUID)
```

But this is an unwieldy line and difficult to understand when viewing the script because it will be broken up over multiple lines. Breaking this function into three separate variables accomplishes the same task with no loss of speed or functionality.

The purpose of this function is to get the custom field ID that the user chose for the condition and try to find that custom field ID in the database. The second half of the block deals with the unlikely possibility that the user entered a custom field ID that does not exist. The system will make a note in the log file that the condition automatically failed because the chosen custom field does not exist, then return *false* for the condition.

```
// look up the LineReading for this reading type

var lineReading = SYS_EVENT.getLine().fetchReading(readingType);

if (lineReading == null)

{

  LogService.error(CAT, "Line did not have reading for readingType code " +
readingType.getCode());

  return(false);}
```

This next block assumes that the previous conditional block was not executed (the user chose a valid custom field). The first line creates a new variable, lineReading, and assigns to that variable the data value found in the chosen custom field for the current transaction line. If there is no data value in this custom field, the system will enter a note in the log file that the credit rule failed because the transaction line had no value in this custom field, and will also return a value of *false* for the condition.

```
var lineStringValue = lineReading.getStringValue();

var retVal = Evaluator.strCompare(lineStringValue, operator, stringValue);

return(retVal);
```

The final block of code for this script assumes that there is a valid string value in the chosen custom field. The system gets this string value and assigns it to another new variable, lineStringValue. The next variable, retVal, calls the Evaluator utility that was opened at the beginning of the script. The system passes the lineStringValue, operator, and stringValue variables into the Evaluator. The Evaluator is what does the actual work of comparing the value in the chosen custom field with the value that the user set up for the condition, and it uses the operator variable to determine how the comparison is to be made. For example, if the user chose "is greater than" for the operator, then the Evaluator checks to see if the custom field value on the transaction line is larger than the value the user chose for the condition.

Finally, the value generated for the retVal variable (either *true* or *false*) is returned as the final result of the condition.

C Recipient Templates Reference

The following credit recipient templates are found in every standard Siebel Incentive Compensation Management installation. The explanations of the JavaScript codes will help guide you in constructing your own customized scripts and templates.

Line Rep of Given Rank

This recipient template is designed to run through the line rep participants listed on a transaction line and choose the one assigned to a rank that matches the rank that the user chooses when setting up the credit rule distribution. For example, if the user specifies a rank of "3" then the system would go through the line reps list on a transaction line and pull out whichever participant had been assigned to rank 3.

```
// RecipLineRepByRank.script

//

// Crediting Recipient Template: Return Credit Recipient Collection containing
indicated Line Rep

//  - Parameters:

//    - Rank

//
```

These lines are each preceded by "//" which indicates that they are comment lines, and are ignored by the system. These comment lines tell you what the script's name is, its function, and what parameters will be used by the script.

```
importClass(Packages.com.againtech.framework.logging.LogService);

importClass(Packages.com.againtech.framework.jndi.JNDIService);

var CAT = "com.againtech.ce.install.script.RecipLineRepByRank.script";

LogService.info(CAT,"Recipient Template: RecipLineRepByRank");
```

This script block is a standard part of all pre-packaged scripts in Siebel Incentive Compensation Management. The first two lines call up standard services that should be included in every script. The next line creates the variable CAT and assigns the script name and location to that variable. The last line writes information to the logging service, so that the Create Credits service can note which recipient template was used during the process.

```
// Execution Context

//
```

```
LogService.info(CAT,"HASHMAP->" + HASHMAP.toString());
```

This line writes additional information to the logging service. The term HASHMAP refers to the mapping between entity codes that users see in Siebel Incentive Compensation Management and the unique ID codes (the UUIDs) that Siebel Incentive Compensation Management automatically assigns to those entity codes. These "hash" codes are normally hidden from users and cannot normally be accessed or referenced directly. In this case, the script gets the "hash" code for the current instance of the Create Credits process and writes it to the log file for future reference.

```
// Input Parameters

var rank = PARAMS.getChild("input_0").getText();

LogService.info(CAT,"rank->" + rank);
```

The second line of this block creates the variable "rank" and assigns to it the text value that the user entered when the recipient template was used in a credit rule. This value will change from one use of the template to another, according to which rank the user wants to assign credits to. The third line logs this value and labels it for ease of reference.

```
// Output Collection

var recipients = new java.util.ArrayList();
```

This line is common to most credit recipient templates. Many recipient templates will return a list of participants or organizations that should receive credit distributions, such as templates that generate credits for all header reps on a transaction. This type of list is handled in JavaScript by creating a variable, in this case "recipients", that is an array of values. For this script it is assumed that only one participant on a transaction line will be assigned to a particular rank, so this array list will contain at most one value when the script ends. In other scripts this array may contain many entries.

```
// Get Participant: Line Rep indicated by rank parameter

var participant = null;

var iter = SYS_EVENT.getLine().getLineReps().iterator();

while (iter.hasNext())

{

       var salesRep = iter.next();

       if (salesRep.getRank().toString().equals(rank))

       { participant = salesRep.getParticipant();

              break;}

}
```

This block contains an iteration process to run through each line rep on a transaction and find the one rep with the rank chosen for the credit rule. The first line creates the "participant" variable and gives it an initial "null" value. The next variable is the iteration variable, which fetches the line reps from the transaction line and contains the iterator operation. The next block runs through each of the line reps. The system gets the first line rep and checks that participant's rank on the transaction line. If it matches the rank chosen for the credit rule, then that participant's ID is written to the "participant" variable and the looping condition stops; the rest of the script then proceeds. If not, then the loop goes back to the transaction line and gets the next line rep and repeats the process. This continues until either one line rep matches the condition or until the system has run through all the line reps without finding a match.

```
// Bail if Line Rep not found

if (participant == null)

{

      LogService.info(CAT,"Can't find Line Rep of Rank: " + rank + " in
   RecipLineRepByRank.");

      return java.util.Collection(recipients);

}
```

In the event that no valid line rep could be found (no one with the proper rank was on the transaction) the system notes in the log file that it couldn't find a valid line rep with the given rank and returns an empty list of participants. This means that no one will receive credits according to this particular distribution.

```
// Add Participant to Collection


recipients.add(participant)



return java.util.Collection(recipients);
```

The last lines of the script are executed if the script found a valid line rep of the chosen rank. This participant's ID is added to the "recipients" array, and this array list is returned as the list of participants that receive credits.

Line Rep of Given Rank's Organization at Given Level

This template is one of the more complex recipient templates found in Siebel Incentive Compensation Management. It will find a line rep of a chosen rank, then find an organization to which that line rep reports at a specific level. This may be the organization that the line rep reports to, or it may be a higher organization to which the rep reports through a chain of organizations. To properly execute this function the script must not only refer to the transaction line but to the organization hierarchy to locate the proper organization at the proper level.

```
   // RecipOrgLineRepByRankAndLevel.script
```

```
//
// Crediting Recipient Template: Return Credit Recipient Collection containing
Organization
// at indicated Level in the hierarchy to which the Organization belongs to which
// indicated Line Rep reports
//  - Parameters:
//    - Rank
//    - org level
//
importClass(Packages.com.againtech.framework.logging.LogService);
importClass(Packages.com.againtech.framework.jndi.JNDIService);


var CAT = "com.againtech.ce.install.script.RecipOrgLineRepByRankAndLevel.script";
LogService.info(CAT,"Recipient Template: RecipOrgLineRepByRankAndLevel");


// Execution Context
//
LogService.info(CAT,"HASHMAP->" + HASHMAP.toString());
```

These opening lines of the script are very similar to the opening lines of the previous script and perform the same functions.

```
// Input Parameters
//
var rank = PARAMS.getChild("input_0").getText();
LogService.info(CAT,"rank->" + rank);
var orgLevelUUID = PARAMS.getChild("entity_0").getText();
LogService.info(CAT, "orgLevelUUID ->" + orgLevelUUID);
```

This block of script declares two variables, each corresponding to one of the variables the user must enter when setting up a distribution (the desired line rep rank and the desired organization hierarchy level). Note that these lines are normally automatically generated when you set up the initial parameters of the condition template, but the system will assign generic names to these variables. The names used in this script (rank and orgLevelUUID) were manually modified so that these variables would be more easily identifiable elsewhere in the script and their function more readily apparent to anyone else that wants to reference the script. For each variable the script also logs the user's chosen values to the log file.

```
// Get Participant: Line Rep indicated by rank parameter

var participant = null;

var iter = SYS_EVENT.getLine().getLineReps().iterator();

while (iter.hasNext())

{

       var salesRep = iter.next();

       if (salesRep.getRank().toString().equals(rank))

       {

              participant = salesRep.getParticipant();

              break;

       }

}


// Bail if Line Rep not found

if (participant == null)

{

       LogService.info(CAT,"Can't find Line Rep of Rank: " + rank + " in
   RecipOrgLineRepByRankAndLevel.");

       return;

}
```

This section of the script is nearly identical to the same block found in the previous script. It runs through the list of line reps and finds the one at the chosen rank. If it can't find such a line rep, the log service is updated to reflect this fact and the script ends, returning a blank list of recipients.

```
// Get Employee Servicer

var employeeServicerHome =
JNDIService.findHome(Packages.com.againtech.ce.employee.EmployeeServicerHome.JNDI_
NAME);
```

```
var employeeServicer = employeeServicerHome.create();


// Get Active EmployeeJobVersion

//   Note argument "String empJobUUID" in getActiveEmployeeJobVersion() is misnamed

//   It's really the "EmployeeUUID"

var employeeJobVersion = null;

employeeJobVersion =
employeeServicer.getActiveEmployeeJobVersion(participant.getUUID());


// Bail if Employee Job not found

//

if (employeeJobVersion == null)

{

      LogService.info(CAT,"Can't find Active Employee Job Version for Line Rep of
   Rank " + rank + " in RecipOrgLineRepByRankAndLevel");

      return;

}
```

This rather long script section actually performs a very simple function – once a valid line rep has been found and the corresponding participant ID has been recorded, this block of script looks through the database to find out if that participant is in an active job. If it can’t locate a current job for the employee, then there cannot be an organization to which the employee reports, and thus there will be no recipients for this distribution.

The first part of this block gets the Employee Servicer, which is a standard tool in Siebel Incentive Compensation Management. The second part of the block actually runs this tool and checks the employee’s records to locate an active job record for that employee. If no active job is found, then the system logs this into the log files and stops the script from executing any further.

```
// Get Organization to which Employee Job reports

//

var organization = employeeJobVersion.getOrganization();

if (organization == null)

{

      LogService.info(CAT,"No Organization for Line Rep of Rank " + rank + " in
   RecipOrgLineRepByRankAndLevel");

      return;
```

```
}
```

This block creates a new variable, “organization”, and assigns to it the organization to which the employee reports directly, according to the employee’s currently active job function. If the employee was not assigned to an organization for this job function, the system notes the error in the log file and returns an empty recipient list.

```
// get the organization at the indicated level of the hierarchy to which this
organization belongs

var hierarchyServicerHome =
JNDIService.findHome(Packages.com.againtech.ce.hierarchy.HierarchyServicerHome.JND
I_NAME);

var hierarchyServicer = hierarchyServicerHome.create();
```

Now the script refers to the Hierarchy Servicer, which will perform two duties in the remainder of the script (see below).

```
// get the hierarchy level object so we can use its code if we need it

var hierarchyLevel = hierarchyServicer.getHierarchyLevel(orgLevelUUID);

if (hierarchyLevel == null)

{

     LogService.info(CAT, "No Hierarchy Level object found for identity [" +
   orgLevelUUID + "]");

     return java.util.Collection(recipients);;

}
```

Here the Hierarchy Servicer performs its first job (getHierarchyLevel). It gets the level that the user chose for the distribution and looks through the hierarchy to see if that level exists in the hierarchy. If the level is not found, the script ends by sending a message to the log file and returning an empty set of recipients. If the level is found, the script proceeds to the next section.

```
var orgAtLevel = null;

var auditDate = new java.util.Date();

orgAtLevel =
hierarchyServicer.getRelatedHierarchyMemberAtLevel(organization.getUUID(),
orgLevelUUID, SYS_PERIOD.getBeginDate(), auditDate);

if (orgAtLevel == null)

{

     LogService.info(CAT, "No organization found for level " +
   hierarchyLevel.getCode() + " in RecipOrgLineRepByRankAndLevel.");

     return java.util.Collection(recipients);

}
```

Now the Hierarchy Servicer performs its second job. First the script creates two new variables that the servicer will need to refer to. The "orgAtLevel" variable will hold the organization that the servicer (hopefully) finds at the appropriate hierarchy level. The "auditDate" variable simply holds the current date. The servicer then looks through the hierarchy level for an organization at the level the user chose for the distribution. It will find a valid organization if, at any time during the current processing period, an organization was put in that level. This allows for the organization hierarchy to change during a period, such as an organization being removed from the specified hierarchy level without a replacement, while still allowing the credit rule to distribute credits to the correct organization.

If the system cannot find an organization in the chosen level for the current period, then the system ends the script by sending a note to the log file and generating an empty list of recipients.

```
// Add Organization at given Level to Collection

var recipients = new java.util.ArrayList();

recipients.add(orgAtLevel);


return java.util.Collection(recipients);
```

Finally, assuming that the script has reached this point (all other conditions have checked out), the system creates the "recipients" array and adds the organization ID that it found through the previous steps. This organization's ID is the only one that will be added to the array and will end up being the sole recipient for this distribution.

D Glossary

Active
The status of any record in Siebel Incentive Compensation Management that is currently available and usable in an incentive plan. See also *Inactive*.

Adjustment
1. A modification made to an original transaction record or credit record to correct data on the original. 2. A modification made to a summarized payout amount to either increase or decrease the payout amount, such as a draw or a cap.

Aggregate
See Cumulate

Aggregate Operator
A mathematical function assigned to a custom field that determines how performance data will be cumulated, if the measure that uses that custom field requires cumulations to be calculated.

Alias
An arbitrary name assigned to a context variable for reference within a calculation formula or a summary formula.

Attribute
Any specific bit of data associated with a record, such as an employee's code in an employee record.

Balance
See Draw Balance

Calculation
Any process performed during the *Plan Calculation* service that performs any kind of numeric or conditional computation on input data.

Calculation Formula
A formula used to calculate payout results for a specific part of an incentive plan. See also *Formula*, *Summary Formula.*

Calculation Result
The final numeric result value of a calculation formula for an individual participant.

Calendar
A series of related and consecutive calendar years.

Calendar Year
One fiscal year of an incentive plan within Siebel Incentive Compensation Management. A calendar year is comprised of one or more periods, and these periods may be grouped into segments.

Cancel
1. An action that cancels the creation or modification of a record. 2. A specific type of adjustment that completely reverses a transaction, transaction line, or transaction line event.

Cap
A limit placed on an employee or on all employees within a plan that determines that maximum incentive payout an employee can earn in any period.

Carry-Forward
A specific type of cap in which amounts that exceed the cap in one period are carried over to the next period and may be paid out in that period.

Ceiling
A limit placed on a draw that determines the maximum amount that an employee may receive solely through draw adjustments.

Channel Partner
A type of participant. A channel partner is usually any business or other third party with which the company has a partnership and through which the company sells products or services.

Child
A term for any organization in a hierarchy that reports to an organization at a higher level, or for any product in a hierarchy that is a sub-product or sub-category of a higher level product. The organization or product at the higher level is called the "parent" of the lower level organization or product.

Closed
A term for a transaction line event that has been credited and for which payouts have been generated, and that has been marked as "closed" by the Close Period service based on this criteria. See also *Open*.

Code
A unique string of letters and/or numbers that identifies a record within the Siebel Incentive Compensation Management database.

Component
One specific calculation step within a calculation or summary formula. The result of one component may be fed into a later component or it may be the end result of that formula.

Condition
A component of a rule that tests one attribute or custom field of a record against a specific value or against another attribute of a record. If the test is successful the condition's result is "true," and if the test fails then the condition's result is "false."

Condition Template
A type of template used to build the selection conditions used in credit rules or plan rules.

Context Attribute
An object that represents a specific data field and that may be referenced in a credit rule or a formula. The context attribute specifically refers to one data field associated with a particular type of data record, such as an employee record.

Credit
A performance data record that tracks actual results for participants or organizations.

Credit Rule
A set of conditions that select specific transaction line event records and generate credit records according to distribution settings.

Credit Rule Set
A set of credit rules that are related in how they select and credit transactions. Transaction lines must match the rule set's header and line profile settings and its event eligibility criteria in order to be passed into the set's credit rules.

Cumulate
The process of summarizing performance data for a calendar segment. The summarization may involve adding data records together, finding an average value amongst data records, and so on.

Cumulation
A setting associated with a measure that determines how performance data will be cumulated. A cumulation setting includes a frequency group, which selects a particular calendar segment type, and a data group, which selects the relevant periods' data within the chosen segment type.

Custom Attribute
Customized data fields that are set up to extend the data that can be recorded for specific types of data records.

Custom Field
Customized data fields that appear on credit, goal, transaction header, and transaction line records. Custom fields build profiles, which in turn determine how these records are structured.

Customer
1. A type of participant in an incentive plan, usually a business or individual that purchases products/services from the company and receives incentives for continued business. 2. A business or individual to which salespeople sell products/services.

Dashboard
The primary screen that users and participants see when they log into Siebel Incentive Compensation Management; the Dashboard is customized for specific types of users and provides them quick access to crucial performance data and overviews.

Data Group
A cumulation setting that determines which periods within a chosen frequency group are relevant for the purposes of that cumulation.

Detail Line
See Transaction Line

Distribution
The part of a credit rule that determines how participants and organizations will receive credit for a transaction. The distribution references a recipient template to determine who receives the credit and also specifies how transaction record data is to be transferred to credit records.

Draw
A financial arrangement between an employee and the company in which the company gives the employee an advance on anticipated future earnings, which usually must be repaid at a specified time.

Draw Balance
The total amount in draw adjustments that an employee has received over any number of periods, and which must be repaid to the company at a specific time.

Draw Ceiling
The maximum amount that an employee can earn in draw adjustments over the course of the calendar year.

Earned Credit
The amount of credit earned for a transaction line based on a rule set's event eligibility criteria; this amount of credit may be further modified for specific distributions.

EIM
Enterprise Incentive Management

Employee
A type of participant that works for the company and receives at least part of his/her total payout from an incentive plan.

Enterprise Unit
The highest level unit in Siebel Incentive Compensation Management that encompasses the entire company, its subsidiaries or holding companies, and its various departments, regions, branches, etc.

ERP
Enterprise Resource Planning

Event
A specific incident in the history of a transaction line. Events are significant because they usually indicate that the transaction is eligible for crediting to participants or organizations.

Event Eligibility
A part of a credit rule set that checks a transaction line's events and allows only those line events that have not been closed to be passed into the set's credit rules.

Event Type
A generic type of event that may apply to any transaction line.

Export
The process of extracting payout data from the Siebel Incentive Compensation Management database and copying the data to an XML file that can be later imported into any payroll or HRMS system.

Formula
A set of components that, when processed in order, determine payout results. Formulas make use of performance data and any data that can be referenced through context attributes; this data is processed through each of the formula's components. There are two types of formulas, Calculation Formulas and Summary Formulas.

Frequency Group
A cumulation setting that chooses a specific calendar segment and allows performance data to be cumulated throughout the periods encompassed by those segments. See also *Data Group*.

Goal
A performance data record that tracks target performance objectives for participants or organizations.

Header
See Transaction Header

Header Participant
A participant listed on a transaction header. Recipient templates often refer to header participants to determine credit distributions.

Hierarchy
A structure that orders organizations or products according to specific levels and relates lower level organizations or products to higher level organizations or products.

Hierarchy Level
A ranking assigned to an organization or product that determines its place relative to other organizations or products within a hierarchy.

If-Then-Else
A calculation component that tests a specified condition and executes a particular calculation if that condition is true, otherwise it executes a different calculation if the condition is false.

Import
The process of extracting data records from an XML file and copying those records to the relevant Siebel Incentive Compensation Management database tables. The XML file is generated by an external software system.

Inactive
The status of any record in Siebel Incentive Compensation Management that is no longer available and that cannot be referenced in any incentive plan. Inactive records cannot be viewed or modified. See also *Active*.

Interpolation
A calculation method used only in step calculations to determine a result value. Interpolation involves locating an input value within a specific range of values, then locating the result value at that same location within a specific range of result values.

JavaScript
A scripting language used to get data from specified sources, perform calculations or conditional tests on that data, and execute specific actions based on the calculated results.

Job
An attribute of employee records that indicates the employee's function or role within the company. Jobs are frequently used in plan rules to determine plan eligibility.

Job History
The complete record of job functions that an employee has held during his or her employment with the company. Jobs within a job history may be used in determining plan eligibility.

Label
The text attached to a specific field within the user interface; this text describes what data is found in that field or that should be entered in that field. Labels may be customized.

Levels
See Hierarchy Levels

Line
See Transaction Line

Line Event
See Transaction Line Event

Line Participant
A participant listed on a transaction line. Recipient templates often refer to a line participant or the list of line participants to determine credit distributions.

Line Type
See Transaction Line Type

Locale
A record that indicates where a user lives or works and the language that user works with.

Matrix
A type of calculation component set up as a table of rows and columns; when used in a formula, the formula uses specific data to look up a row and column and locate the result value where that row and column intersect.

Measure
An object used to track specific types or categories of performance. Measures are associated with every goal and credit record in Siebel Incentive Compensation Management and determine how those records will be cumulated and/or rolled up through the organization.

Open
A term for any transaction line event that has not been "closed" by the Close Period service, meaning that credits and payouts based on that line event have not been finalized.

Open Balance
A specific type of data group in which the performance data from all periods within a frequency group, except the current period's data, are cumulated.

Operating Unit
The second highest type of unit in Siebel Incentive Compensation Management, usually representing a subsidiary, branch, or other division within the extended enterprise.

Organization
Any division, branch, region, or other department within the company. Organizations in Siebel Incentive Compensation Management usually correspond to the organization units within the company's organization structure.

Parent
A term for any organization in a hierarchy to which other organizations report, or any product in a hierarchy that acts as a higher level category for other products or sub-categories of products. The organizations or products that "report to" the parent are called "child" products or organizations.

Participant
A term for any employee, channel partner, or customer that receives payouts from an incentive plan.

Pay When
A setting in credit rules and formulas that determines when a participant will actually receive payouts from a credit record. This setting may indicate immediate payout or it may indicate that payout is to be delayed until a specified period or until a certain number of periods have passed.

Payout
A generic term for actual rewards given to participants, usually in the form of cash but payouts may also be given as prizes, stock options, and so on.

Performance Data
A general term for credit records and goal records, which track actual performance and target performance objectives.

Period
A division of a calendar year that defines the beginning and end of one plan processing cycle.

Plan
A Siebel Incentive Compensation Management plan corresponds to a specific incentive plan used by the company. The plan consists of a plan rule, a set of calculation formulas that are executed for participants that match the plan rule, and summarization instructions for determining final payouts based on the calculation formula results.

Plan Rule
A set of conditions that determine which participants are eligible to be paid out according to the specific plan.

Product Measure
Any measure that is associated with a product. A product's measure is often referenced by a credit rule distribution; this assigns the product's measure to the generated credit records.

Product Rate
See Rate

Profile
A specific set of custom fields that is used to build specific performance and transaction records. Profiles are referenced by transaction types and by measures.

Rate
The commission percentage paid to a sales person or other employee. Specific rates are set up for each product and associated with a rate group.

Rate Group
An attribute that determines which product rate will apply to a particular employee. Rate groups are either assigned directly to employees or they may be assigned indirectly through the employee's assigned job code.

Recipient
Any participant or organization that receives credit from a transaction, according to the distribution settings of credit rules.

Recipient Template
A type of template used to select the participants and/or organizations that will receive credit from a credit rule distribution.

Recovery
The process of retrieving draw balance amounts from an employee, generally by reducing that employee's incentive payout.

Recovery Guarantee
An amount that limits how much of an employee's draw balance may be recovered in any period. The system cannot recover amounts that would reduce the employee's payout below this guaranteed amount.

Return
A specific type of transaction adjustment that reverses specific numeric fields on a transaction line to indicate that the product sold was returned.

Role
A set of privileges that specify which parts of the Siebel Incentive Compensation Management system a user may access and whether the user may create and edit certain types of records or only search for and view such records. Every user is assigned one or more roles.

Rolling Balance
A type of data group that cumulates performance data from the current period and the past x periods, where x is determined by the chosen frequency group.

Rollup
A setting associated with a measure that determines how performance data is rolled up, or copied, from lower level organizations to higher level organizations.

Root Element
The top-most organization or product in a hierarchy to which all other organizations or products report to, either directly or indirectly.

Rounding Rule
A rule assigned to a formula or formula component that dictates how that formula's or component's result should be rounded, if rounding is necessary.

Rule
A general term for credit rules and plan rules. A rule consists of one or more conditions.

Rule Set
See Credit Rule Set

Segment
A division of a calendar year that encompasses one or more periods. Segments are always longer in length than a period. Performance data is usually cumulated across the periods within a segment, and in some plans payouts are not made until the end of a segment.

Service
A general term for any of the processing functions of Siebel Incentive Compensation Management, including import and export, crediting, plan calculation, and period closing.

SFA
Sales Force Automation

Step Calc
A type of calculation component similar to a one-column matrix, with the additional ability to interpolate the result value based on the input value, add multiple result values for the same input value, and perform additional calculations on the result value(s).

Summarization
The process of adding or otherwise summarizing the results of a plan's calculation formulas for the purpose of generating a final payout result for plan participants.

Summary Formula
A formula that specifies how calculation formula results are to be combined to generate final payouts to participants.

Template
A script function written in JavaScript that builds either a selection condition or a credit recipient group. See *Condition Template, Recipient Template.*

Threshold
A calculation component used mainly in sales commission plans that determines commission rates for individual transactions based on the employee's cumulative sales for the period or calendar segment.

To Date
A type of data group that cumulates performance data for all periods within a segment up through the current processing period.

Transaction
A data record that represents any kind of action, such as a sale or contract signing, that contributes to a participant's performance record. A transaction consists of a header and one or more detail lines.

Transaction Event Type
See Event Type

Transaction Header
The portion of a transaction record that records data pertinent to all of the transaction's detail lines, such as the transaction type, the transaction's customer, and specific participants.

Transaction Line
The portion of a transaction record that records specific sales data for one part of that transaction, such as a specific product sold, sales amounts, and specific participants.

Transaction Line Event
An event assigned to a transaction line to indicate that event has occurred for that line.

Transaction Line Type
A category assigned to a transaction line, referenced primarily in credit rule conditions.

Transaction Type
A category assigned to a transaction that determines the transaction's header profile and line profile.

Type
See Transaction Event Type, Transaction Line Type, Transaction Type

Unit
A general term for enterprise units and operating units.

User
Any individual that is authorized to access data within Siebel Incentive Compensation Management. The level of access that a user is permitted is determined by the role or roles assigned to the user.

Variable
See Context Attribute

Working Period
The period in which a user is viewing or modifying records. This may correspond to the current processing period or to a prior period.

XML
Short for Extended Markup Language, a scripting language that allows programmers to define their own HTML properties and attributes. XML is frequently used to format data records so that data can be transmitted between different applications, especially Web-based applications.

Index

S

T

U