Siebel Incentive Compensation Management Configuration Guide

Version 7.8.2, Rev. A

May 2006

ORACLE | SIEBEL

Contents

Chapter 1: What's New in This Release

Chapter 2: Overview of Siebel Incentive Compensation Management Configuration

About Siebel ICM Configuration 17

The ICM Model 17

- Entities 17
- Attributes 19
- Context 20
- Periods and Versions 20

Roadmap for Configuring Siebel ICM 20

Ongoing Processes for Configuring Siebel ICM 21

Chapter 3: Setting Up Access to Siebel Incentive Compensation Management

About ICM Access 23

Process of Setting Up ICM Access 24

- Setting Up the ICM System Administrator 25
- Setting Up an Enterprise Unit 26
- Setting Up Operating Units 28

Chapter 4: Creating the Calendar

About the ICM Calendar 31

Adding a Calendar Year 33

Chapter 5: Setting Up Security

About ICM Security 37

Process of Setting Up ICM Security 38

- Adding Roles and Assigning Privileges 39
- Importing Employee, Customer, and Channel Partner Records 39
- Adding Users and Assigning Roles 40

Chapter 6: Setting Up Currencies and Currency Conversions

About Currencies and Currency Conversions 43

Process of Setting Up Currencies and Currency Conversions 43

- Creating Currencies 44
- Assigning Currencies to an Operating Unit 44
- Importing Currency Exchange Rates 45
- Setting Currency Exchange Rates Manually 45
- Specifying Currency Conversion Options 46

Chapter 7: Defining Profile Attributes and Profiles

About Profile Attributes and Profiles 49

Process of Defining Profile Attributes and Profiles 51

- Defining Profile Attributes 52
- Setting Up a Profile 52

Chapter 8: Setting Up Types

About Setting Up ICM Types 55

Process of Setting Up ICM Types 56

- Setting Up Transaction Types 57
- Setting Up Transaction Line Types 57
- Setting Up Transaction Event Types 58
- Setting Up Incentive Types 59

Chapter 9: Defining Extended and Inherent Attributes

About Extended and Inherent Attributes 61

Defining Extended Attributes 62

Editing Extended Attributes and Inherent Attributes 64

Chapter 10: Modifying Labels

About Labels 67

Modifying Labels 67

Chapter 11: Defining Smart Attributes

About Smart Attributes 69

Defining Smart Attributes 70

Chapter 12: Setting Up Templates

About Templates 73

Setting Up Condition Templates 74

Setting Up Recipient Templates 75

Chapter 13: Setting Up Reference Data

About Reference Data 79

Tasks for Setting Up Reference Data 80

- Setting Up Jobs 80
- Setting Up Salary Grades 82
- Setting Up Cost Centers 83
- Setting Up a Payroll System 83
- Setting Up Locations 84
- Setting Up Channel Segments 84

Chapter 14: Defining Organizations and Organization Hierarchies

About Organizations and Organization Hierarchies 87

Process of Constructing an Organization Hierarchy 88

- Adding Organization Hierarchy Levels 88
- Setting Up Organizations 88
- Setting Up the Organization Hierarchy's Root Element 89
- Adding Child Organizations to the Hierarchy 90

Chapter 15: Defining Products and Product Hierarchies

About Products and Product Hierarchies 91

Process of Constructing a Product Hierarchy 92

- Setting Up Rate Groups 92
- Adding Product Hierarchy Levels 92
- Setting Up Products 93
- Setting Up a Product Hierarchy's Root Element 95
- Adding Child Products to a Product Hierarchy 95

Chapter 16: Setting Up Participants

About Participants 97

Setting Up Employees 98

Process of Changing Employee Job Status 101

- Modifying Employee Job Information 101

Changing an Employee's Job 102

Process of Setting Up Channel Partners 103
Adding Channel Partners 103
Adding Channel Partner Details 104
Adding Channel Partner Contacts 105
Adding Channel Partner Certifications 105

Process of Setting Up Customers 106
Adding Customer Records 106
Adding Customer Details 107
Adding Customer Locations 108
Adding Customer Contacts 109

Chapter 17: Defining Territories and Regions

About Territories and Regions 111

Process of Constructing a Region and Territory Hierarchy 112
Adding Region Hierarchy Levels 112
Adding Regions 113
Adding Employees to a Region 113
Adding Channel Partners to a Region 114
Adding Territories 115
Adding Qualifiers to a Territory 115
Adding Employees to a Territory 116
Adding Channel Partners to a Territory 116
Setting Up the Region and Territory Hierarchy's Root Element 117
Adding Child Regions and Territories to the Hierarchy 117

Chapter 18: Establishing Password Policy

About ICM Password Policy 119

Setting the ICM Password Policy 120

About Configurable Password Properties 120
Hash Method 121
Failed Attempt Limit 121
Failed Attempt Timeframe 122
Lock on Failed Attempt Limit 122
Lock Timeout 122
Password Minimum Age 123
Password Minimum Length 123
Password Strict Syntax 123
Safe Modification 124
Number of Passwords to Remember 124

Filtering Security-Related Log Messages 125

Chapter 19: Setting Up the Publish Service

About the Publish Service 129

Process of Setting Up and Running the Publish Service 129
- Setting Up Informatica 130
- Editing the Informatica Properties 130
- Preparing a Publish Service Configuration File 130
- Running the Publish Service Through the ICM UI 133
- Running the Publish Service in a Service Batch 133

Chapter 20: Managing Services and Service Batches

About Services 135
- Import Services 135
- Export Services 136
- Processing Services 136

Setting the Services Logging Level 138

About Service Batches 139
- Service Batch Framework 139
- Service Launcher 139

Process of Setting Up the Service Batch Framework 140
- Creating a Service Batch with the Service Batch Framework 140
- Importing a Service Batch 141

Process of Setting Up the Service Launcher 141
- Setting Up a Service Launcher Properties File 142
- Creating or Editing a Service Batch File 146

Chapter 21: Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Integration

About Siebel CRM-to-Siebel ICM Integration 149

Process of Running Siebel CRM-to-Siebel ICM Integration 151
- Installing the Integration Workflow into Siebel CRM 151
- Installing the Integration Objects into Siebel CRM 152
- Setting Siebel ICM Configuration Properties 153
- Customizing Siebel ICM to Receive Siebel CRM Data 154
- Extracting Siebel CRM Data with Siebel ICM 157
- Extracting Siebel CRM Data with a Service Batch File 159
- Loading the Siebel CRM Extract into Siebel ICM 159

Reference Implementation Field Mappings 159

About Data Volume and Performance 161

Troubleshooting Siebel CRM-to-Siebel ICM Integration 162

Chapter 22: Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Single Sign-On

About Siebel CRM to Siebel ICM Single Sign-On 165

Process of Configuring Siebel CRM to Siebel ICM Single Sign-On 165

- Identifying Siebel CRM Responsibilities for Siebel ICM Access 166
- Setting Siebel ICM Properties for Siebel CRM Authentication 166
- Mapping Siebel ICM Roles to Siebel CRM Responsibilities 167
- Copying Siebel ICM Web Templates to Siebel CRM 168
- Creating Siebel ICM Screen Tab Objects 168
- Adding an ICM Screen Tab to a Siebel CRM Application 173
- Adding Siebel ICM Screen Access to Responsibilities 173
- Configuring Siebel CRM to Edit Seed Data 173
- Testing the Siebel ICM Tab in Siebel CRM 175

Chapter 23: Setting Up Report Documents

About Report Documents and Actuate Server 177

Process of Setting Up Report Documents with Siebel ICM 177

- Adding an ICM Folder on the Actuate Server 178
- Granting the ICM System User Access to the Actuate Folder 178
- Defining Report Document Security Access 179
- Making Report Documents Available in Siebel ICM 179

Chapter 24: Setting Up Dashboards

About Dashboards 181

Process of Managing Dashboards 181

- Creating a Dashboard 181
- Adding Dashboard Pages 182
- Adding Content to a Dashboard Page 183
- Granting Access to a Dashboard 186

Chapter 25: Removing Deleted Data

About the Reaper 189

Configuring the Reaper to Remove Deleted Records 189

Chapter 26: Removing Obsolete Objects with JVM

About the JVM Garbage Collector 195

Configuring ICM to Optimize Multi-Processor Sun JVM Garbage Collection 196

Chapter 27: Setting Up Debugging and Logging

About Debugging and Log Files 199

Process of Generating Debugging Information 200

- Changing the Logging Configuration 201
- Adjusting the Logging Level in the Application Log File 201
- Generating and Sending the Logs 202

About Appenders and Categories 202

- Appender Definitions for ICM Log Files 202
- Category Definitions for ICM Log Files 205

Appendix A: Smart Attribute Scripts Reference

About Smart Attribute Scripts 209

Example Smart Attribute Script 209

Appendix B: Template Scripts Reference

About Template Scripts 211

Example Condition Template Scripts 211

- Line Product Is Specified Product 211
- Line Field (String) <operator> <value> 213
- Hdr Field (String) <operator> <value> 215

Example Recipient Template Scripts 216

- Line Rep of Given Rank 216
- Line Rep of Given Rank's Organization at Given Level 218

Appendix C: Informatica Transaction-Loading Mappings Reference

About the Transaction Model 223

Process of Mapping the Sales Transaction Loader 224

About Source Data File Formats 225

Informatica Objects for the ICM Transaction Data Model 230

ICM Sales Transaction Data Model 232

Transaction Loader Limitations 233

Best Practice for Informatica Analytics Updates 234

Appendix D: Profiling Siebel Incentive Compensation Management with JProfiler

About JProfiler and ICM 235

Specifying JProfiler Settings for ICM 235

Guidelines for Profiling ICM with JProfiler 236

Appendix E: Open Integration Framework Reference

About ICM's Open Integration Framework 239

- Basic XML Schematics 239
- Profile Attributes and Custom Attributes in XML Files 240
- Dependency-Sorted Import Services 242
- Date and Time Formats 242
- About Translating XML Files Into ICM Format 242

Import Objects Schemas 242

- Employees, Job History, and Jobs 243
- Sales Transactions 246
- Credits 250
- Goals 252
- Products and Product Hierarchy 253
- Customers 255
- Organizations and the Organization Hierarchy 259
- Territories and Territory Hierarchy 260
- Channel Partners 265
- Exchange Tables and Rates 268
- Matrix Expressions 268

Appendix F: XML Service Batch Schema

Index

1 What's New in This Release

What's New in Siebel Incentive Compensation Management Configuration Guide, Version 7.8.2, Rev. A

Table 1 lists changes described in this version of the documentation to support Release 7.8.2 of the software.

Table 1. New Product Features in Siebel Incentive Compensation Management Configuration Guide, Version 7.8.2, Rev. A

Topic	Description
"Modifying Employee Job Information" on page 101	A procedure has been added for modifying employee job information.
"About Configurable Password Properties" on page 120	Configurable password properties descriptions have been updated.
"Filtering Security-Related Log Messages" on page 125	A procedure has been added for filtering security-related messages that are written to Oracle's Siebel Incentive Compensation Management (ICM) product log files.
Chapter 19, "Setting Up the Publish Service"	A new chapter about the Publish Service is added to the guide.
Chapter 25, "Removing Deleted Data"	A new chapter on deleted ICM records removal with the Reaper is added to the guide.
Chapter 26, "Removing Obsolete Objects with JVM"	A new chapter on recommended JVM settings for obsolete items removal is added to the guide.
Chapter 27, "Setting Up Debugging and Logging"	The former "Log Files Reference" appendix has been combined with the "Setting Up ICM Debugging" chapter. The material has been reorganized and updated.
"About Source Data File Formats" on page 225	Descriptions of the Informatica source data mapping file components have been updated.
"Best Practice for Informatica Analytics Updates" on page 234	A best practice topic has been added for configuring Informatica workflow mappings to improve Update Analytics performance.
"Employee Schema" on page 244	A correction has been made to the description of the Employee Schema.
"Managing Operating Unit Exports and Imports" "Service Processes Reference"	A chapter and an appendix have been removed from this guide and moved to *Siebel Incentive Compensation Management Administration Guide*.

Table 1. New Product Features in Siebel Incentive Compensation Management Configuration Guide, Version 7.8.2, Rev. A

Topic	Description
"Participant Snapshot"	Instructions for accessing the Participant Snapshot have been removed from this guide and moved to *Siebel Incentive Compensation Management Administration Guide*.
"Running a Service Batch with the Service Batch Framework" "Running a Service Batch with the Service Launcher"	Instructions for running service batches have been removed from this guide and moved to *Siebel Incentive Compensation Management Administration Guide*.

What's New in Siebel Incentive Compensation Management Configuration Guide, Version 7.8.2

Table 2 lists changes described in this version of the documentation to support Release 7.8.2 of the software.

Table 2. New Product Features in Siebel Incentive Compensation Management Configuration Guide, Version 7.8.2

Topic	Description
"Attributes" on page 19	New tables compare the various types of ICM attributes.
"Setting Up an Enterprise Unit" on page 26	Procedure is updated for new and changed functionality.
"Setting Up Operating Units" on page 28	Procedure is updated for new and changed functionality.
"Importing Currency Exchange Rates" on page 45	The name of the Exchange Rate Import service has been changed to Currency Conversion Rate Import.
"Setting Up Reference Data" Chapter 14, "Defining Organizations and Organization Hierarchies" Chapter 15, "Defining Products and Product Hierarchies" Chapter 16, "Setting Up Participants" Chapter 17, "Defining Territories and Regions"	New chapters have been added to the guide.
Chapter 18, "Establishing Password Policy"	A chapter is added to describe new password policy configuration options.

Table 2. New Product Features in Siebel Incentive Compensation Management Configuration Guide, Version 7.8.2

Topic	Description
"Setting the Services Logging Level" on page 138	The description of errors in service logs has been expanded.
"About the Service Launcher Properties" on page 142	New properties pollInterval and smartPolling have been added to the Service Launcher.
"Managing Operating Unit Exports and Imports"	References to "Operating Unit Sets" were changed to "Migration Sets."
"Creating a Service Batch to Import a Migration Set" on page 175	A new feature allows users to create a service batch from a migration set.
"Hdr Field (String) <operator> <value>" on page 215	Information about the use of the IS LIKE condition in a condition template was added.
Appendix C, "Informatica Transaction-Loading Mappings Reference"	A new appendix on Informatica sales transaction loading mappings is added to the guide.
Appendix D, "Profiling Siebel Incentive Compensation Management with JProfiler"	An appendix was added describing how to set up a third-party performance enhancement utility to be compatible with ICM.
Throughout	The term "credit rule" was changed to "distribution rule." The term "credit rule distribution" was changed to "distribution rule credit."

Additional Changes for Version 7.8.2

The chapters in this guide have been reordered to reflect an overall roadmap for setting up an ICM instance to the point of readiness for compensation administrators to begin constructing compensation plans. Chapters have been reorganized internally to present the sequential process flows of tasks for using each of the application modules.

What's New in Siebel Incentive Compensation Management Configuration Guide, Version ICM 7.8

Table 3 lists changes described in this version of the documentation to support Release ICM 7.8 of the software.

Table 3. New Product Features in Siebel Incentive Compensation Management Configuration Guide, Version ICM 7.8

Topic	Description
Entire book	ICM's application-level menu system has been reorganized.
Chapter 7, "Defining Profile Attributes and Profiles"	*Custom field* has been changed to *profile attribute*.
Chapter 9, "Defining Extended and Inherent Attributes" Chapter 11, "Defining Smart Attributes"	The reference attributes UI has been consolidated. ICM displays the smart attributes, extended attributes, and inherent attributes for each entity on one screen.
"Inherent Attributes" on page 62	The Extended Attributes interface can be used to modify the attributes of *inherent attributes*.
"Inherent Attributes" on page 62 "Editing Extended Attributes and Inherent Attributes" on page 64	*System attribute* has been changed to *inherent attribute*.
Chapter 11, "Defining Smart Attributes" Appendix A, "Smart Attribute Scripts Reference"	*Context attribute* has been changed to *smart attribute*.
"Process of Setting Up the Service Batch Framework" on page 140	You can use the Service Batch Framework to create, import, run, and monitor batches of services.
Chapter 21, "Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Integration"	The chapter about integrating Oracle's Siebel Customer Relationship Management (CRM) product family to Siebel ICM has been reorganized and expanded.
Chapter 22, "Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Single Sign-On"	You can configure Siebel CRM to access Siebel ICM directly with the same sign-on.
"Creating Siebel ICM Screen Tab Objects" on page 168	The procedure for creating Siebel ICM tab objects in a Siebel CRM application has changed. Instead of copying and modifying existing objects, you create new ones.

Table 3. New Product Features in Siebel Incentive Compensation Management Configuration Guide, Version ICM 7.8

Topic	Description
"Managing Operating Unit Exports and Imports"	The masterEntityExportList.properties and operatingUnitEntityExportList.properties files are renamed operatingUnitImportList.properties and operatingUnitExportList.properties, respectively.
Chapter 24, "Setting Up Dashboards"	New Dashboards functionality has been added.
"Adding Content to a Dashboard Page" on page 183	ICM can display reports on a dashboard.
Chapter 27, "Setting Up Debugging and Logging"	Procedures for how to generate and transmit debugging information to ICM have been added to this guide.
"Types of ICM Log Files" on page 199	New locations for JBoss and Websphere log configuration files are specified in the guide. Also, the guide provides a detailed explanation of logging modes.

2 Overview of Siebel Incentive Compensation Management Configuration

This chapter provides an overview of Oracle's Siebel Incentive Compensation Management (ICM) product configuration. It contains the following topics:

- "About Siebel ICM Configuration" on page 17
- "The ICM Model" on page 17
- "Roadmap for Configuring Siebel ICM" on page 20
- "Ongoing Processes for Configuring Siebel ICM" on page 21

About Siebel ICM Configuration

This guide is for system administrators or high-level plan designers who access the core components of ICM to configure the application according to their companies' business practices. This guide covers many of ICM's configuration and customization functions, and discusses some of the back-end processes that govern how it operates.

A companion book, *Siebel Incentive Compensation Management Administration Guide*, is intended for plan administrators and other end users of ICM. It 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 ICM.

The ICM Model

The following topics describe some of the concepts around which Siebel ICM is constructed and how these concepts work together.

- "Entities" on page 17
- "Attributes" on page 19
- "Context" on page 20
- "Periods and Versions" on page 20

Entities

An *entity* is any object in the database, such as an employee history record, calculation formula, or credit rule, that ICM can use for a specific purpose. ICM entities are classified into the following major types:

- **System Entities.** The fundamental database objects in Siebel Incentive Compensation Management. They form the basis for other entities in the database, and without them other entities cannot be created. System entities include enterprise units, operating units, the calendar, service types, users, roles, and locales. They also include customization entities such as profiles, profile attributes, templates, extended attributes, and smart attributes.
- **Incentive Entities.** Objects that Siebel Incentive Compensation Management uses to process incentive plans and calculate earning results. These include distribution rules and credit rules, calculation formulas and formula components, performance measures, and plans. Although you can customize individual incentive entities, you cannot customize the basic structure and usage of an incentive entity.
- **Setup Entities.** Objects other than incentive entities that form a company's incentive plans. These include participants, 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 distribution rule may refer to product data to determine credit distributions. Or, a formula may refer to employee data to determine earning results.

  You can modify setup entities, but they usually remain constant over several processing periods. For example, a company's organization structure generally remains constant over months or years. Occasionally, an organization will be restructured, but this is not a periodic event.

  You can customize setup entities through *extended attributes*, which are additional attributes that extend the types of data tracked for an entity. Extended attributes do not alter the basic types of data, or attributes, that are tracked for that entity.
- **Transaction Entities.** Objects that are either periodically imported or entered into ICM (such as transactions, credits, and goals), or periodically calculated by ICM (such as credits and earning results).

  Like setup entities, transaction entities are referenced by incentive entities during earning calculation. Unlike other entities, each transaction entity is tied to one specific period. This period dependence helps ICM determine, for example, when earnings should be calculated for specific credit records.

  You can customize certain transaction entities through *profiles*. Profiles are custom attributes that determine what data is tracked for transaction entities. At the beginning of period processing, transactions, goals, credits, rollup credits and cumulated credits can be customized with profiles.

  NOTE: At the end of period processing, earnings, summarized earnings, trial payments and payments can be customized with formulas.

  Profiles are reusable across all transaction entities of the same type; for example, a single transaction profile can be used by any credit.

Attributes

An *attribute* describes an entity. Specifically, an attribute is a particular type of data associated with a data record or entity. For example, an employee record is an entity, and the employee's social security number is an attribute of that record. ICM has several types of attributes, which are listed, defined, and compared in Table 4 and Table 5.

Table 4. Attribute Type Definitions

Attribute Type	Comments	Used In
Extended Attribute	Customized field that extends the data included in non-transaction records.	Setup entities. For a complete list, see "Extended Attributes" on page 61.
Inherent Attribute	System attribute that comes with your application.	All entities
Profile Attribute	Customized field that extends the data included in transaction and performance records. Used to build *profiles*, which are used to create transaction records and performance records.	Transaction headers Transaction lines Credit records Goal records
Smart Attribute	Custom script that can retrieve data from fields or other entities. Can also perform computations or derive values. Allows crediting and calculation functions to reference any data entity in the system.	Distribution rules Plan rules Formulas

Table 5. Attribute Type Characteristics

Attribute Type	User Defined?	Creates Field in UI?	Required to Add?	Published to Analytics Database?
Extended Attribute	Yes	Yes	No	Yes
Inherent Attribute	No[1]	Yes	N/A	Yes
Profile Attribute	Yes	Yes	Yes[2]	Yes
Smart Attribute	Yes	No	No	No[3]

1. Inherent attributes associated with setup entities and transaction entities can be edited.

2. Not required by the application, but must be added to support your company's business requirements.

3. A smart attribute's result may be published as a formula reading, if the smart attribute is used in a formula and if that formula variable's result is defined as persistent.

Context

All operations in ICM are performed within a specific *context*. This helps define the scope of an operation or a data entity. Certain setup entities are system-owned; that is, their settings span an entire ICM instance. Other entities are defined within the context of an operating unit. They are applicable only to that operating unit and are not shared or used by any other operating unit, even within the same enterprise unit.

Context defines the scope of the crediting and calculation processes. Credits, for example, are generated within the context of a single transaction event, calendar period, and operating unit. This automatically defines what data is available to the system to create those credits.

Periods and Versions

All ICM functions are performed in the context of a specific calendar *period*. You create a record in a period, and you can modify it in another period. A transaction may be entered in a period. A service is run within a period. This period is the entity's *effective start period*. It tells when that record, in its current form, was first valid in the system.

Versions are related to periods. Versions help keep records of every change or addition made to the database. They allow a user to review the system as it was within a prior period.

When a record is created and marked with an effective start period, that period identifies the first version of that record. If the record is later modified, the period in which the edit was made becomes the new effective start period, and this identifies the second version of the record. At the same time, that period also becomes the *effective end period* of the first version. This may continue indefinitely, because records are continually modified or adjusted.

When multiple changes to a record are made within a single period, ICM considers the last change to be the valid one for that period. When a period is closed, the system captures the version of each record in the database as it is at that moment, and preserves those records for future reference.

Roadmap for Configuring Siebel ICM

This topic lists the stages for configuring Siebel ICM so that the application is fully functional and plan designers can begin to set up compensation plans in the system. The stages are described in specific chapters in this guide, and are followed in this order:

1. Chapter 3, "Setting Up Access to Siebel Incentive Compensation Management"
2. Chapter 4, "Creating the Calendar"
3. Chapter 5, "Setting Up Security"
4. Chapter 6, "Setting Up Currencies and Currency Conversions"
5. Chapter 7, "Defining Profile Attributes and Profiles"
6. Chapter 8, "Setting Up Types"
7. Chapter 9, "Defining Extended and Inherent Attributes"

8 Chapter 10, “Modifying Labels”

9 Chapter 11, “Defining Smart Attributes”

10 Chapter 12, “Setting Up Templates”

11 Chapter 13, “Setting Up Reference Data”

12 Chapter 14, “Defining Organizations and Organization Hierarchies”

13 Chapter 15, “Defining Products and Product Hierarchies”

14 Chapter 16, “Setting Up Participants”

15 Chapter 17, “Defining Territories and Regions”

16 Chapter 18, “Establishing Password Policy”

Ongoing Processes for Configuring Siebel ICM

This topic lists the ongoing operations for maintaining a Siebel ICM instance. The operations are described in specific chapters in this guide, and can be followed in any order, as needed.

- Chapter 19, “Setting Up the Publish Service”
- Chapter 20, “Managing Services and Service Batches”
- Chapter 21, “Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Integration”
- Chapter 22, “Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Single Sign-On”
- Chapter 23, “Setting Up Report Documents”
- Chapter 24, “Setting Up Dashboards”
- Chapter 25, “Removing Deleted Data”
- Chapter 26, “Removing Obsolete Objects with JVM”
- Chapter 27, “Setting Up Debugging and Logging”

3 Setting Up Access to Siebel Incentive Compensation Management

This chapter describes ICM application setup after installation and includes the following topics:

- "About ICM Access" on page 23
- "Process of Setting Up ICM Access" on page 24

About ICM Access

This topic describes how to set up access to the ICM system after it is installed. This topic includes descriptions of enterprise units, operating units, and the system administrator.

Enterprise Units

An *enterprise unit* (EU) represents a company or other business enterprise. It can be either a single entity or an extended organization that encompasses multiple subcompanies and other units. An enterprise unit is the basic building block for your compensation plans. At least one enterprise unit must exist in the system before you can enter or import any other data.

You can set up any number of enterprise units, but the system does not treat these as related units. In other words, data related to one enterprise unit is not shared with any other enterprise unit.

Every enterprise unit must have at least one enterprise unit administrator. This person can modify any data within the enterprise unit and its operating units. Additionally, the enterprise unit administrator is the only user who can add other users to the system and assign them the appropriate security roles. You add enterprise unit administrators when you create the enterprise unit. All other types of users are defined after the enterprise unit has been set up.

Operating Units

An *operating unit* (OU) represents a major division or branch of an enterprise unit. For example, if an enterprise unit is an extended corporation, then each operating unit might represent one company that is part of that larger enterprise.

You must define at least one operating unit for each enterprise unit. Typically, multiple operating units report to the enterprise unit. However, you could establish only one operating unit that reports to an enterprise unit. This could be the case if, for example, a company administers incentive plans for one division and not for others.

An operating unit can report to only one enterprise unit. If a single operating unit actually reports to more than one enterprise unit, then you must set up that operating unit separately within each enterprise unit. The system will treat these operating units as separate entities.

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

Like enterprise units, each operating unit has an *operating unit administrator*. This user can modify any part of the operating unit and its data. This user is usually responsible for maintaining plan details and participant records. Typically, operating unit administrators are *not* the same people as enterprise unit administrators, because enterprise unit administrators have privileges to create other operating units.

Operating Unit-Level Data

Most of the data in ICM is stored at the operating unit level, and is usable only in the operating unit in which the data is created or imported. Such data includes, but is not limited to, the following types:

- 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
- Distribution rules and credit rules
- Transactional data, including sales transactions, credits, earnings, summarized earnings, trial payments, and payments

The System Administrator

The *system administrator* is the user responsible for initially setting up ICM. The system administrator can create enterprise units and assign enterprise unit administrators.

The system administrator role may be performed by the same person who also fills another type of administrator role. If so, that person should have two or more user IDs—one for the system administrator role and a separate one for each other role.

You must set up the system administrator user ID and password before using any other functions in Siebel Incentive Compensation Management, even before starting the ICM application server. This is because the system administrator is the only user who can set up enterprise units and operating units and assign administrator users to those units.

Process of Setting Up ICM Access

This topic describes the process of setting up access to the ICM system after it is installed. This process is composed of the following tasks:

1 “Setting Up the ICM System Administrator” on page 25

2 “Setting Up an Enterprise Unit” on page 26

3 “Setting Up Operating Units” on page 28

Setting Up the ICM System Administrator

Creating the system administrator user ID is a two-step process. First, you generate a secure password for the ID. Second, you set up the ID with a name and the secure password.

The procedure in this topic assumes that Siebel Incentive Compensation Management has been completely installed on one or more server systems.

NOTE: Throughout this guide, <DEPLOYMENT> refers to the network location of the application server with ICM deployed (installed) in it. <STAGING> refers to the user-defined temporary location where configuration of ICM occurs before deployment.

This task is a step in “Process of Setting Up ICM Access” on page 24.

To set up the ICM System Administrator

1 Do one of the following:

- 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 have installed Siebel Incentive Compensation Management on a Solaris or AIX server, log on to this system.

2 Navigate to the <STAGING>/deploy directory and, at the command prompt, enter the following command:

```
ant encrypt-text -Dtext=<USERNAME>
```

Replace <USERNAME> with the user name you want the system administrator to use. The user name can contain only letters and numbers.

The system returns a string of letters and numbers between a pair of square brackets ([ ]). This is the hash equivalent of the user name you entered.

3 Select the string, without the brackets, and copy it.

4 Navigate to the <STAGING>/config directory, open the SiebelICMConfig.xml.in file in a text editor, and locate the following line:

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

At the end of this line, paste the hashed password string returned in Step 2.

5 Navigate to the <STAGING>/deploy directory and, at the command prompt, enter the following command:

```
ant encrypt-text -Dtext=<PASSWORD>
```

Replace <PASSWORD> with the password you want the system administrator to use. The password can contain only letters and numbers.

The system returns a string of letters and numbers between a pair of square brackets ([ ]). This is the hash equivalent of the password you entered.

6 Select the string, without the brackets, and copy it.

7 In the SiebelICMConfig.xml.in file, find the following line:

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

At the end of this line, paste the hashed password string returned in Step 5.

8 Save the SiebelICMConfig.xml.in file and close it.

9 Run the following command:

```
ant deploy-config
```

NOTE: When logging on to ICM subsequently, the system administrator enters the username specified in Step 2 and the password specified in Step 5. The system translates the entered username and password into their hash equivalents, compares those values to the ones in the SiebelICMConfig.xml.in file, and allows access only if the values match.

If a user tries to enter the hash equivalent username and password values from the SiebelICMConfig.xml.in file, the system converts those values into their own hash equivalents. The resulting values do not match the values in the SiebelICMConfig.xml.in file, which prevents a successful logon.

Setting Up an Enterprise Unit

You must set up enterprise units through the *Bootstrap screen,* which is a special system administrator's interface. The following task provides instructions on how to set up an enterprise unit.

This task is a step in "Process of Setting Up ICM Access" on page 24.

To set up an enterprise unit

1 Open a Web browser on your Siebel Incentive Compensation Management application server and navigate to the following address:

http://<HOST>:<PORT>/<APPLICATION NAME>/bootstrap

For <HOST>, substitute the name of the machine where the ICM application server is deployed; for example, sdcdl360i120. For <PORT>, substitute the name of the Web port where the ICM application server is deployed; for example, 8080. For <APPLICATION NAME>, substitute the name of the ICM application; for example, Siebel. To find the correct values for your system, see Table 6.

Table 6. Values in the Bootstrap URL

For This Value:	Go to This File:	See This Property:
<HOST>	<STAGING>/deploy/deploy.default.properties	deploy.app.host
<PORT> (for Tomcat)	<STAGING>/deploy/jboss3.0/appserver.properties	deploy.port.tomcat.HttpConnector
<PORT> (for WebSphere)	<STAGING>/deploy/websphere5.1/appserver.properties	deploy.port.websphere.HttpConnector
<APPLICATION NAME>	<STAGING>/deploy/deploy.default.properties	deploy.app.name

2 At the Bootstrap logon screen, complete the necessary fields. Some fields are described in the following table.

Field	Comments
User Name	User name of the system administrator. Required value is mark.
Password	Password of the system administrator. Required value is anil.
Language	Default language to display on this Enterprise Unit's user interface.
Country	Default country for which this Enterprise Unit's user interface will be localized.

3 Click Login.

4 On the Create Enterprise Unit form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the enterprise unit.
Description	Full name of the enterprise unit.
Beginning Period	First day of the earliest calendar year in all operating units this enterprise unit will contain.
Status	Leave this field set to Active.

Field	Comments
Functional Currency	Currency in which the enterprise unit typically processes transactions and earnings.
Administrator User Name	User name for this enterprise unit's administrator. Can contain any combination of letters and numbers. Cannot contain spaces (use the underscore character (_) instead) or any non-alphanumeric characters. **NOTE:** The EU administrator may be the same person as the system administrator, but it is recommended that each administrator role have a separate user name and password.
Password	Enterprise unit administrator's password. Must be at least six characters long and contain both numbers and letters. Can contain only alphanumeric characters or the underscore character. Cannot contain spaces.
Confirm Password	Re-enter the same password you entered in the Password field.

5 Click Save to complete the Enterprise Unit setup.

6 The system redisplays the Bootstrap logon screen.

Setting Up Operating Units

After setting up the enterprise unit, you can set up multiple operating units (OUs) within it. As with enterprise units, you set up operating units manually through the *Bootstrap screen,* a special system administrator's interface. After creating operating units, you can export and import them between different ICM environments.

Adding Operating Units Manually

To set up an operating unit manually through the Bootstrap screen, follow this procedure.

This task is a step in "Process of Setting Up ICM Access" on page 24.

To add an operating unit manually

1 Open a Web browser on your Siebel Incentive Compensation Management application server and navigate to the following address:

`http://<HOST>:<PORT>/<APPLICATION NAME>/bootstrap`

For <HOST>, substitute the name of the machine where the ICM application server is deployed; for example, sdcdl360i120. For <PORT>, substitute the name of the Web port where the ICM application server is deployed; for example, 8080. For <APPLICATION NAME>, substitute the name of the ICM application; for example, Siebel. To find the correct values for your system, see Table 6 on page 27.

2 At the Bootstrap logon screen, enter the Enterprise Unit administrator User Name and Password you specified in "Setting Up an Enterprise Unit" on page 26 and click Login.

3 On the Create Operating Unit form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the operating unit.
Description	Full name of the operating unit.
Beginning Period	Beginning effective date for this operating unit. **CAUTION:** This field determines the earliest date of the operating unit's calendar. Specify a date that corresponds with the beginning of the company's or branch's calendar, *not* the operating unit's creation date.
Analytics Code	Code that differentiates this operating unit from all others when publishing transaction data to the Analytics database.
Status	Leave this field set to Active.
Functional Currency	Currency in which the operating unit typically processes transactions and earnings. **NOTE:** The operating unit's functional currency may be the same currency as the enterprise unit, or it may be different. When processing transactions, credits, and earnings, the operating unit's functional currency overrides the enterprise unit's functional currency.
Administrator User Name	User name for this operating unit's administrator. Can contain any combination of letters and numbers. Cannot contain spaces (use the underscore character (_) instead) or any non-alphanumeric characters. **NOTE:** The OU administrator may be the same person as the system administrator, but it is recommended that each administrator role have a separate user name and password.

Field	Comments
Password	Operating unit administrator's password. Must be at least six characters long and contain both numbers and letters. Can contain only alphanumeric characters or the underscore character. Cannot contain spaces.
Confirm Password	Re-enter the same password you entered in the Password field.

4 Click Save to compete the Operating Unit setup.

The system displays the Siebel ICM logon screen.

5 To launch the new OU, at the Siebel ICM logon screen, enter the Operating Unit administrator User Name and Password you specified in Step 3 and click Login.

ICM starts up and displays an interface for the new OU. The system prompts you to create a calendar year. Go to "Adding a Calendar Year" on page 33.

Exporting and Importing Operating Units

ICM allows you to move data between a development or test environment and a production environment by exporting an operating unit data set from the originating environment and then importing it into the target environment. For information about exporting and importing operating unit data sets, see the chapter on exporting and importing operating units in *Siebel Incentive Compensation Management Administration Guide*.

NOTE: OU import and export is backward compatible. In other words, you can export an OU from a previous ICM version and import it into your current version.

4 Creating the Calendar

This chapter describes the ICM calendar and provides instructions for setting it up. It contains the following topics:

- "About the ICM Calendar" on page 31
- "Adding a Calendar Year" on page 33

About the ICM Calendar

The ICM *calendar* defines the processing periods of your company's incentive plans. The first thing you do after creating a new operating unit is to set up its calendar. When you launch ICM and log on for the first time as that operating unit's administrator, the system detects that the OU has no calendar and automatically opens the calendar year creation screen. You must create a calendar for the OU before the system allows any other functions.

Calendar Years

An ICM calendar is composed of one or more *calendar years*. A calendar year can correspond to a standard calendar year or a fiscal year, as defined by your company's business practices. The calendar itself has no definitive end, so you can add more calendar years indefinitely. Once you initiate the calendar and set up one calendar year, any user with the proper privileges can add future calendar years at any time.

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 number of periods that is different from the prior calendar, or with customized segment types incorporated. For more information about periods, see "Calendar Periods" on page 31. For more information about segment types, see "Calendar Segments" on page 32.

Calendar Periods

Each calendar year contains one or more *periods*. A period is defined by a start date and an end date, and contains all dates in between. Whenever the system refers to a specific date, it looks up the period that corresponds to that date. For example, a transaction line has one or more associated events, each marked by a specific date. ICM matches these dates with the calendar year's periods to determine whether the events occurred in the current period, a prior period, or a period that has not yet been processed.

The number of periods in your calendar year corresponds to how frequently you need to process your incentive plans. Most plans are calculated monthly, so most calendar years contain twelve periods. Other plans are calculated quarterly (four periods), semiannually (two periods), or annually (one period). You can also define custom calendar years that have any 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 is the period number within its own calendar year, while the absolute period number is the period number in relation to all other period numbers in all other calendar years.

For example, suppose your calendar follows a twelve-period pattern, and you are setting up its third calendar year. Each period in that calendar year has a relative period number of 1 through 12, and also an absolute period number of 25 through 36. Relative period 1 corresponds to absolute period 25, relative period 2 to absolute period 26, and so on.

NOTE: Periods in a calendar year can contain different numbers of days. For example, suppose you divide your calendar into 12 periods, each corresponding to a month. In this case, each period begins and ends on the first and last days of each month, regardless of whether that month has 28, 29, 30, or 31 days.

Calendar Segments

Every calendar year has one or more *segments*. A segment is defined by a start period and an end period, and contains all periods in between. For example, in a calendar that follows a twelve-period pattern, a quarter segment corresponds to a set of three month-long periods, and every calendar year contains four quarter segments.

Segment types denote different kinds of segments. For example, the quarter segment allows a calendar year to be divided into four quarter segments, while the annual segment type allows a calendar year to contain only one segment, spanning the entire year.

Any number of segment types can be associated with each calendar year. The period type of the calendar year automatically defines the segment type of the calendar year's periods. You can add other segment types as well. 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 add a *quarter* segment type, each quarter segment containing three monthly periods. Additionally, you could add an *annual* segment type, consisting of all twelve monthly periods.

You can also define *custom segment types*. Custom segment types can be created for calendars that do not follow the standard 365 or 366 day pattern, or for regular calendars. For example, you could create a *trimester* segment type that contains three segments for one year, each composed of four monthly periods.

NOTE: Segments in a calendar year can have different numbers of periods. For example, suppose your calendar has 12 periods, each corresponding to a calendar month, and you want to divide your calendar year into five segments. You could assign three periods to each of the first three segments, two periods to the fourth segment, and one period to the fifth segment.

Custom and Standard Calendar Configurations

In some situations, you may need to mix standard and custom calendar configurations in the same calendar year or across calendar years. For example, if the number of periods changes from one year to the next, Earn When formula options will be affected.

Consider a case of semiannual bonus calculations. Suppose that Fiscal Year 2004 has 12 calendar periods, matching a standard monthly calendar. A semiannual bonus is credited in periods 6 and 12 (effectively June 2004 and December 2004). Fiscal Year 2005, however, has 24 calendar periods. If left unadjusted, the formulas that calculate semiannual bonuses will yield payments in March 2005 and June 2005 (periods 6 and 12 in the 2005 setup). To keep the semiannual bonus schedule, the Earn When formulas must be adjusted for Fiscal Year 2005 to yield payments in periods 12 and 24.

For more information about Earn When calculations, see the chapter on calculation formulas in *Siebel Incentive Compensation Management Administration Guide*.

Adding a 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.

To define a custom segment, you decide how many segments of that type will be fitted into a calendar year, and then assign that type a unique name.

To add a calendar year

1 Navigate to Configure > Calendar.

2 Click the New Calendar Year link.

NOTE: If you are opening an OU for the first time, the system skips Step 1 and Step 2, and displays the Basic Information form.

3 In the Basic Information form, enter a code for the new calendar year in the Code field.

4 Select one of the following:

- **Standard Calendar Year.** Select if the year follows a 365/366-day standard calendar, and all the periods and segments will be regular. Continue with "To set up a standard calendar year" on page 33.
- **Custom Calendar Year.** Select if the year does not follow a standard 365/366-day standard calendar. Continue with "To set up a custom calendar year" on page 35.

To set up a standard calendar year

1 Complete the fields under Standard Calendar Year. Some fields are described in the following table.

Field	Comments
Start Date	Day, month, and year to begin the calendar year.
Period Segment Type	Period type. Determines how many periods will be in the calendar year.

2 Click Next.

The Periods screen shows each period in terms of relative period number, absolute period number, and the date ranges in each period.

3 If desired, change the start and end dates associated with one or more periods.

You can modify the Start Date of any period except the first and the End Date of any period except the last.

NOTE: The end date of every period must be one day before the start date of the next period.

4 Click Save.

The Segments screen shows the segments that have been created for your calendar. The selected period type is automatically a segment type. Any segments that are longer than the period segment type are automatically generated by the system.

5 Do one of the following:

- If your calendar year does not require custom segment types, click Save to complete the calendar year.
- If your calendar year requires custom segment types, continue with "To create custom segment types for a standard calendar year" on page 34.

To create custom segment types for a standard calendar year

1 On the Segments screen, do one of the following:

- To create a new segment type for this calendar, go to Step 2.
- To use custom segment types that are already defined in another calendar year, go to Step 5.

2 Click the New Custom Segment Type link.

3 On the Custom Segment Type Information screen, complete the fields. Some fields are described in the following table.

Field	Comments
Segments Per Year	Total number of segments you want in the calendar year.
Segment Type Code	Identifying code for this segment type.

4 Click Next.

This redisplays the Segments screen.

5 In the drop-down list by the Calendar Segments heading bar, select the segment type and click Add.

The system tries to fit the selected segment type into the calendar year and distributes periods to each segment. The Define Segments screen displays the results. The Define Segments section displays the segments for this segment type, and the calendar periods associated with each segment.

6 If desired, change the calendar periods associated with one or more segments.

You can change the First Period for any segment except the first, and the Last Period for any segment except the last.

7 Click Save.

8 Repeat Step 1 through Step 7 for each custom segment type you want to add to the calendar year.

9 Click Save to complete the calendar year.

To set up a custom calendar year

1 Complete the fields under Custom Calendar Year. Some fields are described in the following table.

Field	Comments
Start Date	Day, month, and year to begin the calendar year.
End Date	Day, month, and year to end the calendar year.
Period Segment Type	Segment type. Determines how many segments will be in the calendar year. Do one of the following: ■ Select an existing custom segment type. Go to Step 4. ■ If no custom segment types have been defined for other calendar years that you want to use in this calendar year, select New Custom Segment Type. Go to Step 2.

2 Click Next.

3 On the Custom Segment Type Information screen, complete the fields. Some fields are described in the following table.

Field	Comments
Segments Per Year	Total number of segments you want in the calendar year.
Segment Type Code	Identifying code for this segment type.

4 Click Next.

The Periods screen shows each period in terms of relative period number, absolute period number, and the date ranges in each period.

5 If desired, change the start and end dates associated with one or more periods.

You can modify the Start Date of any period except the first and the End Date of any period except the last.

NOTE: The end date of every period must be one day before the start date of the next period.

6 Click Save.

The Segments screen shows the segments that have been created for your calendar year, beginning with the selected custom segment type.

7 Repeat Step 1 through Step 6 for each custom segment type you want to add to the calendar year.

8 Click Save to complete the calendar year.

5 Setting Up Security

This chapter describes security settings in ICM and includes the following topics:

- "About ICM Security" on page 37
- "Process of Setting Up ICM Security" on page 38

About ICM Security

This topic describes users, privileges, and roles in ICM. It also explains the relationships between these concepts.

Users

A *user* is any person who must access one or more functions in ICM. A user is identified by a unique user name and password, which she must enter to log on to ICM. Users are generally administrators (employees who create, edit, and review data records and modify incentive plans) or participants (employees, channel partners, or customers who view reports and analyses). Users gain access to data and functions in a specific operating unit through the assignment of a role in that operating unit.

NOTE: Not every participant has to be a user. Your incentive plan may include participants who should not have access to the system. Similarly, administrators may also be plan participants.

Operating unit administrators can view all user records, even for those users who do not have privileges to access the data in their operating units. One user may receive access to several operating units by receiving role assignments from several operating unit administrators. This allows a user to access multiple operating units with a single login name and password.

One user, the generic System Administrator, is already set up with every ICM installation. This allows access to the system during initial setup.

CAUTION: To prevent other users from accidentally gaining access to the system through this user name, it is highly recommended that the system administrator change the generic System Administrator user's settings before adding any other users to the system. The system administrator should assign his user a unique password that only he knows.

Privileges and Privilege Categories

ICM includes multiple types of *privileges* that define access rights to various parts of the system. Privileges apply to *privilege categories*. A privilege category encompasses a specific function or data category within ICM, such as Employees or Formulas. For most privilege categories, you can assign Create/Edit privileges or Search/View privileges to a role. By assigning privileges to a role, you give those privileges to any user assigned to that role.

The types of privileges are as follows:

- **Create.** Allows a user to create new records for the specified category of data, such as products or employees. For most categories, the Create privilege is combined with the Edit privilege.
- **Edit.** Allows a user to edit and delete existing records for the specified category of data. For most categories, the Edit privilege is combined with the Create privilege.
- **Search.** Allows a user to perform search queries within the specified category of data, such as products or formulas. The Search privilege is always combined with the View privilege.
- **View.** Allows a user to open and display records. The View privilege is almost always combined with the Search privilege.
- **Assign.** Allows a user to assign roles to other users. For example, the system administrator typically can assign the Enterprise Unit Administrator role or the System Administrator role to other users.
- **Revoke.** Allows a user to remove assigned roles from other users. Usually, revocation of roles is done only when users change job functions.
- **Adjust.** Allows a user to amend transaction, credit, and earning records. This includes 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.

Roles

A *role* is a set of privileges that specify which areas of your application a user can access, and the access rights they have to those areas. To gain access to ICM, a user must be assigned one or more roles.

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

Some roles, such as Enterprise Unit Administrator and Operating Unit Administrator, are already defined in ICM as part of the initial configuration. These roles have privileges set up, and all you need to do is assign the roles to the appropriate users. In addition, you can define new roles to fit the security policies for your system and database.

Process of Setting Up ICM Security

The process of setting up ICM security consists of the following tasks:

1. “Adding Roles and Assigning Privileges” on page 39
2. “Importing Employee, Customer, and Channel Partner Records” on page 39
3. “Adding Users and Assigning Roles” on page 40

Adding Roles and Assigning Privileges

This procedure describes how to create a role and assign privileges to that role.

This task is a step in "Process of Setting Up ICM Security" on page 38.

To add a role and assign privileges

1 Navigate to Configure > Roles.

2 Click the New Role link.

3 On the Role Information screen, complete the fields. Some fields are described in the following table.

Field	Comments
Name	Unique name for the role.
Copy Privileges from Role	Select a role to copy its privileges into this role, or select None to add all privileges to this role manually.
Operating Unit Administrator	Select if this role is being set up for operating unit administrators. **NOTE:** This check box appears only if you have the privilege in your role to assign operating unit administrator roles. The chief system administrator has this privilege, but other users might not.

4 Click Save.

5 On the View Role screen, click the Edit button next to any privilege category to assign new privileges or to change existing privileges for that privilege category.

6 On the Edit Privileges screen for the privilege category, select the check box for each privilege you want to assign to the role, or clear it to remove that privilege from the role. To assign all privileges in a subcategory, select the All check box for that subcategory.

NOTE: In most cases, if you assign a category's Create/Edit privilege to a role, you must also assign that category's Search/View privilege as well. Otherwise, end users will not be able to search for or view records for editing.

7 Click Save to add the selected privileges to the role.

8 Repeat Step 5 through Step 7 for each category for which you want to add privileges to this role.

Importing Employee, Customer, and Channel Partner Records

After you have set up roles, you can import records for potential users from the systems that track your company's employees, channel partners, and customers.

This task is a step in "Process of Setting Up ICM Security" on page 38.

To import employee, customer, and channel partner records

1. Export the necessary employee, customer, and channel partner records from your company's third-party systems.
2. Make sure the exported XML files are in the correct format to be imported into ICM. For information about the XML import file format, see Appendix E, "Open Integration Framework Reference."
3. Navigate to Master Control > Import & Export Services.
4. Click one of the following links:
   - To import employees, click the Employee Import service link.
   - To import customers, click the Customer Import service link.
   - To import channel partners, click the Channel Partner Import service link.
5. On the Service Launcher screen, click Launch Service.
6. In the Import File field, navigate to the directory where the exchange rates XML file resides, and select the file.
7. Click Import.
8. To import another set of records for potential users, repeat Step 3 through Step 7.
9. After the service has finished processing, you can do the following:
   - To view employees, navigate to Organization > Employees.
   - To view customers, navigate to Organization > Customers.
   - To view channel partners, navigate to Organization > Channel Partners.

Adding Users and Assigning Roles

This procedure describes how to create a user in ICM and assign a role to that user.

This task is a step in "Process of Setting Up ICM Security" on page 38.

To add a user and assign a role

1. Navigate to Configure > Users.
2. Click the New User link.
3. On the User Information screen, according to the type of user you are setting up, do one of the following:

- **Participant.** Select Participant User and complete the fields immediately below it. Some fields are described in the following table.

Field	Comments
Type	Type of user.
Participant ID	Participant's identifying code. You can enter it manually, or select it from the participant IDs in the system.

- **Non-participant.** Select Non-Participant User and complete the fields immediately underneath.

4 Click Next.

5 On the Account Information screen, complete the fields. Some fields are described in the following table.

Field	Comments
User Name	Unique identifier for this user. Can contain only alphanumeric characters and the underscore (_) character. Cannot contain spaces.
Password	User's password. Must be at least six characters long. Can contain only alphanumeric characters and the underscore (_) character. Cannot contain spaces.
Confirm Password	Re-enter the same password you entered in the Password field.
Account Status	Verify that the value is set to Active. Users with Inactive status will not be able to access the system.

6 Click Next.

7 In the drop-down list on the Roles heading bar, select a role and then click Add Role.

Repeat this step for each role you want to assign to this user.

8 Click Save to complete the user record.

6 Setting Up Currencies and Currency Conversions

This chapter describes setting up currencies in the ICM system and it includes the following topics:

- "About Currencies and Currency Conversions" on page 43
- "Process of Setting Up Currencies and Currency Conversions" on page 43

About Currencies and Currency Conversions

ICM can process transactions in multiple currencies. The system can convert values such as transactions, credits, calculations, and earnings that are recorded in one currency into equivalent values in any other currency. This allows, for example, a transaction made in Canadian dollars to be credited and paid in U.S. dollars, or the other way around.

ICM classifies currencies in several ways. The currency that the Operating Unit uses and in which it calculates earnings is the *functional currency*. The currency of record for an incoming transaction is the *transaction currency*. The transaction currency affects how the currency attributes in the header and line items are converted to other currencies. The currency in which payments are displayed and made to a participant is the *participant currency*. The participant currency can be defined for each participant. All these currencies may be different from each other. An Operating Unit can have only one functional currency, but may have any number of transaction currencies and participant currencies.

The functional currency is the main currency used in processing. When processing transactions and calculating results, ICM converts the other currencies to and from the functional currency. For example, consider the case of a Canadian company with a branch sales office in Britain that covers the company's Europe region. The functional currency across the Operating Unit (company) is Canadian dollars. A sales representative at the branch sales office may specify her participant currency as British pounds. Many of her sales may be paid in a transaction currency of Euros. In this case, ICM converts the Euro transaction proceeds to Canadian dollars, calculates the sales representative's earnings in that currency, and then converts the earnings to British pounds for payment to the sales representative.

Process of Setting Up Currencies and Currency Conversions

The process of setting up currencies consists of the following tasks:

1. "Creating Currencies" on page 44
2. "Assigning Currencies to an Operating Unit" on page 44
3. Setting up currency exchange rates by doing one or both of the following:

- "Importing Currency Exchange Rates" on page 45
- "Setting Currency Exchange Rates Manually" on page 45

4 "Specifying Currency Conversion Options" on page 46

Creating Currencies

To allow ICM to perform currency conversions, your system must have Operating Unit-level records for all the currencies that may need to be converted one to another. Records for most currencies come with the system. If you need to create additional currency records, follow this procedure.

This task is a step in "Process of Setting Up Currencies and Currency Conversions" on page 43.

To create currencies

1 Navigate to Configure > Currency Codes.

2 Click the Add Currency Code link.

3 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Identifying code for the currency. It is recommended that you use the standard global three-character currency code, which is usually the same as the ISO 4217 standard. 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 Canada's two-character Internet code (CA) plus a one-character currency designator (D).
Name	Name of the currency—for example, Canadian Dollar.
Symbol	Symbol for the currency. For example, for U.S. Dollars, the symbol is $.

4 Click Save.

Assigning Currencies to an Operating Unit

To make currency conversions possible, you assign every possible functional, transaction, and participant currency to your Operating Unit. You must assign at least one currency to each operating unit.

This task is a step in "Process of Setting Up Currencies and Currency Conversions" on page 43.

To assign a currency to an Operating Unit

1. Make sure you are working in the operating unit to which you want to assign currencies.
2. Navigate to Configure > Unit Currencies.
3. In the drop-down list, select a currency to assign to this Operating Unit.
4. Click the Add Currency icon.
5. Repeat Step 3 and Step 4 for each currency you want to assign to this operating unit.

Importing Currency Exchange Rates

After currency setup is complete, you can import multiple currency exchange rates. You must import exchange rates for every day or period needed for currency conversions, so the system has a basis for conversion calculations. You import the exchange rate information in the form of XML files. You may need to perform this task daily or periodically to keep the system up to date with current exchange rates.

This task is a step in "Process of Setting Up Currencies and Currency Conversions" on page 43.

To import currency exchange rates

1. Navigate to Master Control > Import & Export Services.
2. Click the Currency Conversion Rate Import service link.
3. On the Service Launcher screen, click Launch Service.
4. In the Import File field, navigate to the directory where the exchange rates XML file resides, select the file, and click Open.
5. Click Import.
6. After the service has finished processing, do one of the following:
   - To see the exchange rates for today, navigate to Configure > Daily Currency Rates.
   - To see the exchange rates for the current period, navigate to Configure > Period Currency Rates.

   NOTE: You can also use these screens to edit currency exchange rates that have been imported.

Setting Currency Exchange Rates Manually

After currency setup is complete, you can create currency exchange rates manually, one at a time. You can create daily or period exchange rates. You must create exchange rates for every day or period needed for currency conversions, so the system has a basis for conversion calculations.

NOTE: In the procedure that follows, "participant currency" refers to "transaction currency" as well. If the participant currency and the transaction currency are different, you assign both as operating unit currencies, and then create a period currency conversion rate or a daily currency conversion rate separately for each transaction currency and each participant currency.

This task is a step in "Process of Setting Up Currencies and Currency Conversions" on page 43.

To create a daily currency exchange rate manually

1. Navigate to Configure > Daily Currency Rates.
2. Click the New Daily Currency Rate link.
3. In the Day Date field, enter the calendar day for which this exchange rate will be valid.
4. Complete the other fields, as necessary. Some fields are described in Table 7.

Table 7. Currency Rate Fields

Field	Comments
Functional Currency	Currency in which transactions occur.
Participant Currency	Currency in which participant is to be paid.
Rate, From Functional Currency	Rate for converting from the functional currency to the participant currency.
Rate, To Functional Currency	Rate for converting from the participant currency to the functional currency.

5. Click Save.

To create a period currency exchange rate manually

1. Make sure you are in the working period for which this exchange rate will be valid.
2. Navigate to Configure > Period Currency Rates.
3. Click the New Period Currency Rates link.
4. Complete the necessary fields. Some fields are described in Table 7 on page 46.
5. Click Save.

Specifying Currency Conversion Options

ICM's currency conversion options affect how the system converts transaction currency amounts into functional currency equivalents to generate credit record currency amounts.

This task is a step in "Process of Setting Up Currencies and Currency Conversions" on page 43.

To change currency conversion options

1. Navigate to Configure > Currency Conversion Options.
2. Click the Edit icon.

3 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Use Currency Conversion Rate if Supplied on Transaction	Determines which currency conversion rates to use when converting transaction amounts into functional amounts for credit records. ■ **Selected.** Uses the currency conversion rates specified in transaction headers. If ICM does not find a conversion rate in a transaction header, then it uses the system conversion rates. ■ **Cleared.** Uses the system currency conversion rates.
Transaction Currency Conversion	Determines which date's system currency conversion rates to use when converting transaction amounts into functional amounts for credit records. ■ **Use Daily Rate for Transaction Header Date.** Uses the daily system rates. Determines the specific date by each transaction header date. ■ **Use Daily Rate for Transaction Line Date.** Uses the daily system rates. Determines the specific date by each transaction line date. ■ **Use Daily Rate for Transaction Event Date.** Uses the daily system rates. Determines the specific date by each transaction event date. ■ **Use Period Rate for Transaction Header Date.** Uses the period system rates. Determines the specific date by each transaction header date. ■ **Use Period Rate for Transaction Line Date.** Uses the period system rates. Determines the specific date by each transaction line date. ■ **Use Period Rate for Transaction Event Date.** Uses the period system rates. Determines the specific date by each transaction event date.

4 Click Save.

7 Defining Profile Attributes and Profiles

This chapter describes profile attributes and profiles, and explains how to define them. It contains the following topics:

- "About Profile Attributes and Profiles" on page 49
- "Defining Profile Attributes" on page 52

About Profile Attributes and Profiles

This topic describes profile attributes and profiles.

Profile Attributes

Profile attributes are data fields on some transaction entities that are specially adapted to your company and its incentive plans. You can define profile attributes for transaction headers, transaction lines, credit records, and goal records. You *must* set up profile attributes in every ICM installation. Without profile attributes, transactions and performance records will not contain any information beyond their identifying codes. Profile attributes are used to build *profiles*, which are the basis for transaction types and measures. These are used to create transaction records and performance records.

You can reuse profile attributes across multiple profiles of the same type or different types; in other words, you do not need to define the same profile attributes separately for each profile. For example, your company's transaction records may require a data field to hold the total sales amount of the transaction. This same field might be used in credit records as well, so that the sales amount on a transaction translates to the sales amount on a credit. This field could also be used in goal profiles, providing a direct link between the goal set for sales and the actual sales made.

NOTE: Earning records are also transaction entities, but you cannot create profile attributes or profiles for them. Instead, the system automatically generates the required profile attributes for each result record during earning calculation.

Profile attributes differ from other types of attributes in several ways. For a listing and comparison of the various types of attributes in ICM, see "Attributes" on page 19.

Data Types

A profile attribute can have any one of several data types. The data types that define what sort of data a profile attribute can hold are as follows:

- **String.** This data is treated as text, whether the data field contains letters, numbers, or other characters (such as underscores). For example, if a string contains only numbers, the system still treats it as text.

- **Number.** These fields may only contain numbers, because the system treats the data as numbers that can be used mathematically within formulas.
- **Currency.** This is a special type of number, 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, patterned to match the system's functional date format. For example, U.S. dates are stored in MM/DD/YYYY format (for example, 11/10/2005 for November 10, 2005).
- **Boolean.** These fields handle Yes-No or True-False conditions. Typically, these fields appear as check boxes. Thus, if the check box is selected, the field will read True; if it is cleared, the field will read False.

Aggregate Operators

Aggregate operators tell the system how to cumulate and roll up the data in profile attributes when you run the Cumulate Service or the Rollup Service. For more information about services, see Chapter 20, "Managing Services and Service Batches."

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

The available aggregate operators are as follows:

- **SUM.** Adds together all values for this profile attribute. Available for data types of Currency, Number, or Boolean.
- **MIN.** Finds the minimum value among the values for this field. For numeric and currency fields, finds the smallest value in a range of records. For date fields, finds the record with the earliest date. Available for data types of Currency, Number, Date, or Boolean.
- **MAX.** Finds the maximum value among the values for this field. For numeric and currency fields, finds the largest value in a range of records. For date fields, finds the record with the latest date. Available for data types of Currency, Number, Date, or Boolean.
- **COUNT.** Counts how many records containing this field are in the system, regardless of the values in the fields. Available for any data type.

You can associate only one aggregate operator with each profile attribute. To associate more than one operator with the data in a profile attribute, you must create two profile attributes, each with its own operator. For example, suppose you need to cumulate sales amounts for credit records, and you have to show both the sum of sales amounts and the maximum sales amount for all records. In this case, you would create two profile attributes, Sales_Amt_SUM and Sales_Amt_MAX, and associate the appropriate operator with each one. You would include both profile attributes in the credit profile, and set up distribution rule credits to copy the same value from transaction lines to each field.

Reversible Fields

The Reversible setting indicates that any numeric or currency values stored in that field can be reversed in a credit adjustment. This setting applies only to profile attributes used in credit profiles.

Returnable Fields

The Returnable setting indicates that any numeric or currency values stored in that field can be adjusted automatically when a transaction line is returned. This setting applies only to profile attributes used in transaction line profiles.

Profiles

A *profile* is a set of profile attributes. Each transaction entity type has a corresponding profile type. The types of profiles are transaction header profiles, transaction line profiles, credit profiles, and goal profiles. Any number of profiles of the same type can be created within an operating unit.

Profiles are referenced by transaction header types, line types, and measures. These types and measures are associated with specific transaction and performance records. Thus, the associated profiles determine which profile attributes appear in those records.

For example, suppose that the credit profile Sales Credits contains the profile attributes Sales_Amount, Quantity, and Net_Attainment. A measure called Direct Sales 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, these credit records will contain the Sales_Amount, Quantity, and Net_Attainment profile attributes.

In another example, suppose that the transaction line profile Standard Sales Detail contains the profile attributes Sales_Amount, Quantity, Profit_Margin, and Net_Profit_Amount. A transaction line type called Standard Sale 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 the standard data fields for all transaction lines, these transaction lines will contain the Sales_Amount, Quantity, Profit_Margin, and Net_Profit_Amount profile attributes.

Any number of profiles can include the same profile attribute. Thus, transaction line profiles may contain some of the same fields as credit profiles, credit profiles may share some fields with goal profiles, and so on.

Process of Defining Profile Attributes and Profiles

This topic describes the process of defining profile attributes and creating profiles. This process consists of the following tasks:

1. “Defining Profile Attributes” on page 52
2. “Setting Up a Profile” on page 52

Defining Profile Attributes

Use this procedure to define profile attributes.

NOTE: You can delete profile attributes up until they are used in processing. After that, you cannot remove them from the system.

This task is a step in "Process of Defining Profile Attributes and Profiles" on page 51.

To define profile attributes

1. Navigate to Configure > Profile Attributes.
2. Click the New Profile Attribute link.
3. Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for this profile attribute.
Data Type	Type of data that can be entered in this profile attribute.
Aggregate Operator	Operator that will cumulate and roll up data for this profile attribute. If this is a non-cumulative data field, or if you will not be using the rollup feature, leave this field marked as None.
Returnable for transactions	If this profile attribute will be used in transaction profiles, select to make this field editable on a transaction return.
Reversible for credits	If this profile attribute will be used in credit profiles, select to reverse this field when the system generates a reversing credit for a transaction adjustment, return, or cancellation that has generated credits.
Description	Description of how this field will be used.

4. Click Save.

Setting Up a Profile

Follow this procedure to set up a profile.

This task is a step in "Process of Defining Profile Attributes and Profiles" on page 51.

To set up a profile

1. Navigate to Configure > Profiles.
2. Click the Add Profile icon for the type of profile you want to add.

3 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for this profile.
Description	Description of the profile.

4 Click Save.

5 To add profile attributes to the profile, do the following:

a In the Profile Attributes drop-down list, select a profile attribute code.

b To require end users to fill in this profile attribute for every record that is created with this profile, select the Attribute is Required check box. To allow users to leave this profile attribute blank, leave the check box cleared.

c Click the Add Profile Attribute icon.

d Reorder the profile attributes by clicking the up and down arrows in the Move column.

NOTE: The profile attribute at the top of the list is the Principal Field, which is displayed on the Search screens.

8 Setting Up Types

This chapter describes setting up types in ICM and includes the following topics:

- "About Setting Up ICM Types" on page 55
- "Process of Setting Up ICM Types" on page 56

About Setting Up ICM Types

When configuring ICM, you set up transaction types, transaction line types, transaction event types, and incentive types.

NOTE: Because transaction types, transaction line types, and transaction event types are versioned entities, it is recommended that you create them in the first working period of your company calendar's fiscal year.

Transaction types, transaction line types, and transaction event types are associated with credit rules. For information about credit rules, see the chapter on distribution rules and credit rules in *Siebel Incentive Compensation Management Administration Guide*.

Transaction Types

Every transaction in Siebel Incentive Compensation Management is assigned a *transaction type*. The transaction type determines which transaction header profile and transaction line profile will be used to generate profile attributes for the transaction's header and detail lines.

Transaction types are also used as eligibility criteria by credit rules. Every credit rule is assigned a header profile and a line profile. Transaction types are also assigned a header profile and a line profile, so the credit rule checks for matching profiles. Any transaction type that uses both of the credit rule profiles becomes a valid transaction type for that credit rule. Thus, any transaction associated with that transaction type can be passed into the credit rule and into its individual distribution rules.

Transactions types with line profiles that differ from the credit rule 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, the transaction is blocked from that credit rule and the transaction lines are not credited according to any rules in that credit rule.

Transaction Line Types

A transaction line can be assigned a *transaction line type*. If you need a different set of fields mapped to the credit for that type of transaction line, the transaction line type can determine the transaction line profile for that line. The transaction line type is primarily referenced by distribution rules and can be used to determine distribution rule eligibility for transaction lines.

NOTE: When you use a profile associated with the transaction line type and a profile associated with a transaction type, a unique credit rule may be required.

Transaction Event Types

A transaction line is associated with one or more events. An *event* identifies when something significant happens in a transaction, such as when a product is ordered or shipped, or when a contract is signed. *Transaction event types* define the kinds of events that are significant for your company's transactions. When a transaction is 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 can be assigned to new or existing transactions, if all events are of different types.

Events also determine how much credit is generated for a transaction line. After determining which events on a line are open, the system uses the transaction type to determine whether the transaction qualifies for the credit rule. If it does, the system looks for the open events on the event type list for that transaction type for that credit rule.

Each event type has an associated percentage value. When the system generates credits for an event, the system multiplies that event's credit values by its event type's percentage value to determine the amount of credit that participants receive. This allows participants to earn partial credit on transactions, ultimately earning 100% credit when all events have been entered and processed. Alternatively, participants can earn full credit on a transaction for every qualifying event, allowing multiple credits to be generated from a single transaction line.

Incentive Types

Incentive types are the types of earnings your participants receive. You can use incentive types in formulas to identify different portions of earnings being calculated.

Process of Setting Up ICM Types

The process of setting up types in ICM consists of the following tasks:

1. "Setting Up Transaction Types" on page 57
2. "Setting Up Transaction Line Types" on page 57
3. "Setting Up Transaction Event Types" on page 58
4. "Setting Up Incentive Types" on page 59

Setting Up Transaction Types

You set up transaction types in the Transaction Types view.

This task is a step in "Process of Setting Up ICM Types" on page 56.

To set up a transaction type

1. Navigate to Transactions > Transaction Types.
2. Click the New Transaction Type link.
3. Complete the fields. Some fields are described in the following table.

Field	Comments
Code	Identifying code for the transaction type.
Transaction Header Profile	Profile the system uses to generate profile attributes for the transaction header.
Transaction Line Profile	Profile the system uses to generate profile attributes for the transaction's detail lines.
Default Processing Option	Determines how the system processes transaction events created in a closed period, as follows: ■ **Retroactively.** The event will be processed in the closed period by the retroactive services. Any resulting credits, cumulated credits, and earnings will be generated in the closed period and carried forward through subsequent periods to generate trial payment adjustments in the first open period. ■ **First Open Period.** The event will be processed in the first open period by the non-retroactive services.

4. Click Save to complete the transaction type.

Setting Up Transaction Line Types

Follow this procedure to set up transaction line types in ICM.

This task is a step in "Process of Setting Up ICM Types" on page 56.

To set up a transaction line type

1. Navigate to Transactions > Transaction Line Types.
2. Click the New Transaction Line Type link.

3 Complete the fields. Some fields are described in the following table.

Field	Comments
Code	Identifying code for the transaction line type.
Transaction Line Profile	Profile the system uses to generate profile attributes for the transaction lines. **NOTE:** The ability to set profiles for transaction line types allows transaction lines of different types to contain different profile attributes. If you do not assign a profile to a transaction line type, the system determines the transaction line's profile from the item type.

4 Click Save to complete the transaction line type.

Setting Up Transaction Event Types

Follow this procedure to set up transaction event types in ICM.

This task is a step in "Process of Setting Up ICM Types" on page 56.

To set up a transaction event type

1 Navigate to Transactions > Transaction Event Types.

2 Click the New Transaction Event Type link.

3 Complete the fields. Some fields are described in the following table.

Field	Comments
Code	Identifying code for the transaction event type.
Report Attribute	Identifies this transaction event type for use in reporting. You can use this field to group multiple transaction event types on a single report. For example, suppose you want to include eventTypeA, eventTypeB, and eventTypeC on a report titled ABC Event Types. On each of these transaction event types, set the Report Attribute field as ABC Event Type, and then write the report to look for that attribute.
Qualifies for Crediting	Select if this transaction event type will be used for credit rule qualification.

4 Click Save to complete the transaction event type.

Setting Up Incentive Types

Follow this procedure to set up incentive types in ICM.

This task is a step in "Process of Setting Up ICM Types" on page 56.

To set up an incentive type

1. Navigate to Plan & Payment > Incentive Types.
2. Click the New Incentive Type link.
3. Complete the fields. Some fields are described in the following table.

Field	Comments
Code	Identifying code for the incentive type.
Name	Full name of the incentive type.
Represents money	Select if earnings with this incentive type will be paid to the participant in money. Clear if earnings with this incentive type will be in some form other than money.

4. Click Save to complete the incentive type.

9 Defining Extended and Inherent Attributes

This chapter describes extended attributes and explains how to define them. It also describes inherent attributes and explains how to modify them. This chapter contains the following topics:

- "About Extended and Inherent Attributes" on page 61
- "Defining Extended Attributes" on page 62
- "Editing Extended Attributes and Inherent Attributes" on page 64

About Extended and Inherent Attributes

This topic describes extended attributes and inherent attributes.

Extended Attributes

Extended attributes are attributes that are customized according to the needs of your company. They can be added to setup entities and their corresponding screens, and extend the data that can be recorded for these entities. They appear, from an end user's perspective, as data fields in the ICM user interface. Extended attributes are defined exclusively for the data tables and user interface screens they modify, and cannot be reused elsewhere in the application.

Extended attributes are typically used when an entity's inherent attributes do not cover every aspect of that entity, and additional attributes are needed. Extended attributes are defined at the operating unit level. Thus, extended attributes defined within one operating unit cannot be used by other operating units.

Extended attributes can be referenced by rules to determine distribution rule eligibility or plan eligibility conditions. For example, an extended attribute for employees might be used to determine which plan an employee qualifies for. An extended attribute for products might be used to determine whether a transaction that included a specific product is eligible for a distribution rule.

You can create extended attributes for the following types of records:

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

Extended attributes can be strings (text), numbers, or dates. For an extended attribute of any data type, you can 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 cannot proceed until that field is filled in.

Each extended attribute of any data type also has a *display type*, which governs how the data field will appear in the user interface. The display type may also constrain entries to a list of values or to a range of values.

Extended attributes differ from other types of attributes in several ways. For a listing and comparison of the various types of attributes in ICM, see "Attributes" on page 19.

Inherent Attributes

Inherent attributes are the standard default attributes that come with your ICM application. The inherent attributes of incentive entities and system entities cannot be customized, added to, modified, or removed. The inherent attributes of setup entities and transaction entities can be customized.

You can use the Extended Attributes interface to modify inherent attributes. This can be useful for changes such as constraining certain fields to a list of values or a range of 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.

Inherent attributes differ from other types of attributes in several ways. For a listing and comparison of the various types of attributes in ICM, see "Attributes" on page 19.

Defining Extended Attributes

Use these procedures to define Extended Attributes.

To create an extended attribute

1. Navigate to Configure > Reference Attributes.
2. Click the link for the entity (record type) that includes the extended attribute you want to define.
3. Click the Add Extended Attribute icon.
4. In the New Name field, enter a name for this extended attribute.

   This will be the label for the extended attribute field in the user interface.
5. Click Next to continue.

6 Complete the fields, as needed. Some fields are described in the following table.

Field	Comments
String, Number, or Date	Determines the data type for the extended attribute.
Default Value (for String, Number, Date)	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.
Fill Existing Entities with this Value	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 applies only to existing data records and does not apply to new records.
Attribute is Required	Select if an end user must specify a value for this field before the system can save the record.
Publish to Analytics	Select to write this field to the Analytics database.

7 Click Next to continue.

8 On the Value Constraints screen, select the applicable display type. The display types are listed in the following table.

Display Type	Comments	Available for Data Types
None	Displays the field as a text field or a text area. There are no constraints on the values entered in the field, except that the values must match the field's data type.	All
Defined List of Acceptable Values	Displays the field as a drop-down list, from which end users must select one of the list values. If you select this option, you must define the list of valid values.	All
Range of Acceptable Values	Displays the field as a text field. Entries must fall between, or be equal to, the end points of a range. In the adjoining fields, enter the beginning and end of the range.	Numeric
Value Must Match Regular Expression	Constrains entries to a pattern of text or numbers. The expression specifies the pattern, consisting of specific alphanumeric characters mixed with special characters that stand for any alphanumeric character. The expression validates the field entries. In the adjoining field, enter the expression.	All

9 Click Next to continue and do one of the following:

- If you selected Defined List of Acceptable Values in Step 8, continue with "To define the attribute as a list of values" on page 64.
- If you selected an option other than Defined List of Acceptable Values in Step 8, continue with "To define the attribute as an entry field" on page 64.

To define the attribute as a list of values

1. On the New or Existing List screen, select Create New List to create a new list of values, and click Next.
2. In the Value field, enter the first value that you want to appear on the list.
3. Click Add to List to add the value to the list.
4. Repeat Step 2 and Step 3 for each additional value that you want to appear on the list.
5. Click Next to complete the attribute.

To define the attribute as an entry field

1. On the Choose Display Type screen, select how to display the attribute entry field in the user interface. Selections are described in the following table.

Field	Comments
Text Field (single line)	A single line in which text can be entered.
Text Area (larger scrolling box)	A large box in which multiple lines of text can be entered. Text areas are commonly used for comments or descriptions where an end user may need to enter long strings of text. They are also useful for scripting.

2. Click Next to complete the attribute.

Editing Extended Attributes and Inherent Attributes

Use this procedure to edit extended attributes and inherent attributes.

To edit an extended or inherent attribute

1. Navigate to Configure > Reference Attributes.
2. Click the link for the entity (record type) that includes the attribute you want to edit.
3. Do one of the following:
   - In the Inherent Attributes list, find the inherent attribute you want to edit and click its Add Constraint icon. Go to Step 5.

- In the Extended Attributes list, find the extended attribute you want to edit and click its View icon.

4 In the View Extended Attribute screen, click the Edit icon.

5 Edit the fields, as needed. Some fields are described in the following table.

Field	Comments
String, Number, or Date	Determines the data type for the attribute. **CAUTION:** If records that use the attribute have been added, changing its data type may invalidate previous entries to the field. For example, if you change the data type from String to Number, entries that contain non-numeric characters will no longer be valid for this attribute field. However, you can change from Number or Date to String without invalidating existing entries.
Default Value (for String, Number, Date)	Value to be automatically filled in the field if an end user does not enter or select another value. **NOTE:** If you change this field, the new default value will only affect new record entries. It will not affect any previously entered records that have another value or no value.
Attribute is Required	Select if an end user must specify a value for this field before the system can save the record.
Publish to Analytics	Select to write this field to the Analytics database.

6 Click Next to continue.

7 On the Value Constraints screen, select the applicable display type. The display types are listed in the following table.

Display Type	Comments	Available For
None	Displays the field as a text field or a text area. There are no constraints on the values entered in the field, except that the values must match the field's data type.	All data types
Defined List of Acceptable Values	Displays the field as a drop-down list, from which end users must select one of the list values. If you select this option, you must define the list of valid values.	All data types
Range of Acceptable Values	Displays the field as a text field. Entries must fall between, or be equal to, the end points of a range. In the adjoining fields, enter the beginning and end of the range.	Numeric only
Value Must Match Regular Expression	Constrains entries to a pattern of text or numbers. The expression specifies the pattern, consisting of specific alphanumeric characters mixed with special characters that stand for any alphanumeric character. The expression validates the field entries. In the adjoining field, enter the expression.	All data types

CAUTION: Changing the attribute's display type may invalidate existing entries for some records. For example, if the field previously had a display type None and you assign it the display type Defined List of Acceptable Values, any records with entries that do not appear on the list you create will become invalid.

8 Click Next to continue and do one of the following:

- If you selected Defined List of Acceptable Values in Step 7, continue with "To define the attribute as a list of values" on page 64.
- If you selected an option other than Defined List of Acceptable Values in Step 7, continue with "To define the attribute as an entry field" on page 64.

10 Modifying Labels

This chapter describes labels in ICM and how to modify them. It contains the following topics:

- "About Labels" on page 67
- "Modifying Labels" on page 67

About Labels

Labels identify data fields, screen names, screen sections, links, buttons, and error messages throughout ICM. These labels are text strings that are rendered in the appropriate style for display on the screen. *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.

You can modify most labels to accommodate your company's business practices and terminology. For example, suppose you want to remind your users, when adding new locations, that they should enter a Location ID number of exactly seven digits. For the Add and Edit page-specific label for Locations, you would change `Location ID#` to `Location ID# (must be 7 digits)`.

Modifying Labels

Follow this procedure to modify the labels in ICM's user interface.

To modify labels

1. Navigate to Configure > Labels.
2. In the Select Group list, click the label group link for the label you want to modify.

   The Label values pop-up list for the group you selected appears.
3. Click the link for the label values category that includes the label you want to modify.
4. Locate the label you want to change, and do one of the following:
   - Use the Find and Replace fields at the top of the screen to find and replace label names with new values of your choice.
   - In the Current Value column for the label, click the field to make it active and type in a new value of your choice.
5. If you make a mistake, click the Cancel Edit icon for that label to revert to the original label.
6. Repeat Step 4 and Step 5 for each label in this label values category that you want to change.
7. Click Save to complete your changes.

11 Defining Smart Attributes

This chapter describes smart attributes and explains how to define them. It contains the following topics:

- "About Smart Attributes" on page 69
- "Defining Smart Attributes" on page 70

About Smart Attributes

A *smart attribute* is a custom script that can retrieve data from fields or from other entities, and can manipulate that data by performing computations or deriving values. Smart attributes allow you to reference any data entity in the system from within distribution rules, plan rules, and formulas, by bringing additional data into *context.* Context defines the scope of crediting and calculation functions and determines what is automatically known to the system during each of these processes. Additionally, a smart attribute can perform calculations on the data it retrieves.

A *JavaScript component*, a script written in JavaScript code, provides the smart attribute with its functionality. This script tells the system which data table to get data from, how to look up the correct record, and what calculations are necessary to retrieve the data. For example, a simple script could look up the date on which an employee started his job. A more complex script could compute how many days the employee has held his current job by looking up the employee's job start date, retrieving the current date from a system calendar, and calculating the difference between the two.

CAUTION: Do not try to set up a smart attribute unless you are familiar with using JavaScript. The smart attributes that your company needs to run its incentive plans should have been set up during the initial ICM implementation. If you need to create new attributes and you require assistance, contact your Oracle Technical Support representative or see Appendix A, "Smart Attribute Scripts Reference" for an example of a smart attribute script.

Smart 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 by that OU. If a smart attribute is assigned to a specific OU, it will become part of the OU export. After the OU export, it can be imported into a modeling scenario or into another ICM instance. For more information about OU exports and imports, see the chapter on exporting and importing operating units in *Siebel Incentive Compensation Management Administration Guide*.

Smart attributes differ from other types of attributes in several ways. For a listing and comparison of the various types of attributes in ICM, see "Attributes" on page 19.

Example: Smart Attribute for Distribution Rule

During crediting, each credit is created in the context of one transaction event and the current calendar period. Thus, when the system processes a transaction event through a distribution rule, it can refer to any information found on the transaction event or the transaction header. Additionally, in case the rule's conditions or distributions are period-dependent, the system knows what the current period is. Any data outside this scope is not within the context, so the process cannot reference it. For instance, the product sold on a transaction line is within the context of the transaction event, but details about that product are not within that context.

To bring such additional information within the context, you could set up a smart attribute for the Sales Crediting service that references the Product table and a data field on that table, such as the product's unit price or default rate. Then, when setting up the distribution rule, you could set up conditions with the smart attribute that select transaction records based on the price or rate of the product sold.

Example: Smart Attribute for Calculation Formula

Without smart attributes, calculation formulas can only refer to performance record data for a specific employee or the results of other formula components. A calculation formula could reference credit and goal records for an employee, but it could not look up any information about that employee, such as her job function or the rate group to which she belongs.

To bring such additional information within the context, you could create a smart attribute that refers to an Employee data table and to one of the data fields on that table, such as the employee's assigned job code. In any component of the calculation formula, you could then refer to this smart attribute and use the referenced job code in whatever manner is required.

Defining Smart Attributes

You create a smart attribute by creating a smart attribute record in ICM, and provide its functionality by writing a JavaScript component and attaching it to the record.

To define a smart attribute

1. Navigate to Configure > Smart Attributes.
2. Click the New Smart Attribute link.

3 Complete the fields, as needed. Some fields are described in the following table.

Field	Comments
Name	Name that will be referenced by rules or formulas that include this smart attribute. **NOTE:** Because smart attributes are assigned to specific services, you can use identical names for multiple smart attributes if you assign them to different services. However, you cannot assign multiple smart attributes with identical names to the same service.
Make available during this service	Service to which this smart attribute will be assigned. Smart attributes cannot be shared across services. However, you can create smart attributes that perform the same function but are assigned to different services.
Data Type	Tells ICM what kind of data to return when it runs the script. Options are: ■ **Number.** A numeric result. **NOTE:** The return type of number is a double. **CAUTION:** A common mistake in creating JavaScript components is to generate a result that looks like a number but is formatted as a text string. ■ **Boolean.** A True or False result. ■ **Currency.** A number in currency format. ■ **Date.** A group of characters in date format; for example, MM/DD/YYYY for U.S. dates. ■ **String.** Any text string of alphanumeric characters.
Category	User-defined classification for this smart attribute. You can enter one of the standard categories that come with ICM—for example, Employee, Product, or Employee Job. Alternatively, you can make up your own.

Field	Comments
System Owned	Select to make this smart attribute available to all operating units. Clear to make this smart attribute specific for this operating unit. **NOTE:** System-owned Smart Attributes are not included in Operating Unit exports. Non-system-owned Smart Attributes are included in Operating Unit exports. They are created in the target generating unit when the export set is imported into another Operating Unit. For inclusion in or exclusion from OU exports, this field's initial setting overrides any subsequent modifications.
Script	JavaScript component for the smart attribute. For an example of a smart attribute script, see Appendix A, "Smart Attribute Scripts Reference."

4 Click Save to complete the smart attribute.

12 Setting Up Templates

This chapter describes templates and their uses in ICM. It contains the following topics:

- "About Templates" on page 73
- "Setting Up Condition Templates" on page 74
- "Setting Up Recipient Templates" on page 75

About Templates

A *template* is a script function written in JavaScript that builds either a selection condition (condition template) or a credit recipient group (recipient template). For more information about template scripts, see Appendix B, "Template Scripts Reference."

Templates 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 by that OU. If a template is assigned to a specific OU, it becomes part of the OU export. After the OU export, it can be imported into a modeling scenario or into another ICM instance. For more information about OU exports and imports, see the chapter on exporting and importing operating units in *Siebel Incentive Compensation Management Administration Guide*.

ICM includes several basic condition templates and recipient templates. These templates cover many of the most common conditions used in rules. You can use these templates without modification.

Condition Templates

Condition templates are used to build the selection conditions in distribution rules and earning calculation eligibility rules. The template specifies what data to examine, what to compare that data to, and what eligibility conditions the data must meet. A condition template can be set up for either plan rules or distribution rules, and becomes available to all plan rules or distribution rules.

You can set up customized condition templates to handle more complex conditions than ICM handles by default. For example, distribution rules and plan rules normally consider all conditions to be joined by AND functions, meaning that all conditions must return a value of True. You could create a condition template that tests multiple conditions and returns a value of True if any one of those conditions is satisfied. This effectively provides a distribution rule or plan rule with an OR condition.

Condition templates can also use IS LIKE conditions. For more information, see "Hdr Field (String) <operator> <value>" on page 215.

Recipient Templates

A *recipient template* groups together a set of participants and organizations to create distribution rule credits. The recipient template contains a set of conditions that selects specific employees, channel partners, customers, organizations, and so on. The template can then be referenced from a distribution rule. When ICM generates credits according to that rule, the rule determines which participants or organizations fit the conditions of the recipient template, and generates credits for each one that passes the test. Each recipient receives the same amount of credit. Essentially, the system copies one credit record among all the listed recipients.

You can define multiple sets of recipient templates. You can call multiple templates from one distribution rule by creating one distribution for each recipient template.

Setting Up Condition Templates

Follow this procedure to set up condition templates.

To set up condition templates

1. Navigate to Plan & Payment > Condition Templates.
2. Click the New Condition Template link.
3. Complete the fields, as needed. Some fields are described in the following table.

Field	Comments
Code	Unique ID for the template. This code should describe the template's purpose so that users can easily select the correct template for distribution rules or plan rules.
Name	Full name for the template. This name should fully describe what the template accomplishes.
Make available during this service	Select one of the following: ■ **PlanEligibilityService.** Associates the template with plan rules, which use conditions to test participants for eligibility. ■ **SalesCreditingService.** Associates the template with distribution rules, which use conditions to test transactions and transaction lines for eligibility.
System Owned	Select to make this template available to all operating units. Clear to make this template specific for the current operating unit.

4. Click Next to continue.
5. To add a description, do the following:
   a. Click Descriptive Text.

b In the Enter Text Below field, enter a description of the template.

Describe exactly what the condition will do. This description is what users will see when they add this condition template to a plan rule or distribution rule.

c Click Add.

6 To add an entity, do the following:

a Click Entity Type.

Entities are standard data fields on transactions or participant records. They may also be data fields that are automatically available for a particular service, such as the current calendar period.

b In the Select Type field, select an entity type for the template.

c Click Add.

7 To add other elements to the template, do the following:

- To insert a numeric input field, click Numeric Input.
- To insert a text input field, click Text Input.
- To insert a drop-down list, click List.
- To insert an operator selection list, click Operator.

  This includes the standard condition operators (less than, equal to, not equal to, and so on).

8 If you make a mistake, click the Remove Last Line button to remove the last element you added to the template string.

9 Click Next to continue.

The Condition Template: JavaScript form displays the JavaScript code generated by your element selections.

10 In the Condition Template: JavaScript field, enter additional lines to complete the script. These lines may define, for instance, what the system should do with its input values, what conditions will return a value of True, and what conditions will return a value of False.

For examples of condition template scripts and tips on setting up new scripts, see "Example Condition Template Scripts" on page 211.

11 Click Save to complete the template.

Setting Up Recipient Templates

Follow this procedure to set up recipient templates.

To set up recipient templates

1 Navigate to Sales Crediting > Credit Recipient Templates.

2 Click the New Credit Recipient Template link.

3 Complete the fields, as needed. Some fields are described in the following table.

Field	Comments
Code	Unique ID for the template. This code should describe the template's purpose so that users can easily select the correct template for distribution rule credits.
Name	Full name for the template. This name should fully describe what the template accomplishes.
System Owned	Select to make this template available to all operating units. Clear to make this template specific for the current operating unit.

NOTE: The only available service for this template type is the SalesCreditingService.

4 Click Next to continue.

5 To add a description, do the following:

a Click Descriptive Text.

b In the Enter Text Below field, enter a description of the template.

Describe exactly how recipients will be selected. This description is what users will see when they add this recipient template to a distribution.

c Click Add.

6 To add an entity, do the following:

a Click Entity Type.

Entities are standard data fields on transactions or participant records. They may also be data fields that are automatically available for a particular service, such as the current calendar period.

b In the Select Type field, select an entity type for the template.

c Click Add.

7 To add other elements to the template, do the following:

- To insert a numeric input field, click Numeric Input.
- To insert a text input field, click Text Input.
- To insert a drop-down list, click List.
- To insert an operator selection list, click Operator.

  This includes the standard condition operators (less than, equal to, not equal to, and so on).

8 If you made a mistake, click the Remove Last Line button to remove the last element you added to the template string.

9 Click Next to continue.

The Credit Recipient Template: JavaScript form displays the JavaScript code generated by your element selections.

10 In the Credit Recipient Template: JavaScript field, enter additional lines to complete the script.

These lines may define, for instance, what the system should do with its input values and how to select recipients according to those values. For examples of recipient template scripts and tips on setting up new scripts, see "Example Recipient Template Scripts" on page 216.

11 Click Save to complete the template.

13 Setting Up Reference Data

This chapter describes reference data and includes the following topics:

- "About Reference Data" on page 79
- "Tasks for Setting Up Reference Data" on page 80

About Reference Data

Reference data refers to the principal entities in Siebel ICM whose parameters define each participant. Reference data includes the following entities:

- Channel Segments
- Cost Centers
- Job Codes
- Locations
- Payroll Systems
- Salary Grades

Channel Segments

Channel segments represent the various channels through which your company sells its products—for example, direct sales, channel partner or VAR sales, inside sales, and so on. You assign a customer a channel segment to indicate how products are sold to that customer. A channel partner is assigned a channel segment to indicate the type of business relationship that exists between that partner and your company. Channel segments are rarely employed in ICM, so using them is optional.

Cost Centers

Accounting systems use cost centers to track expenditures across the extended enterprise. Cost centers in Siebel ICM typically correspond to cost centers in payroll and general ledger systems. Cost centers are rarely employed in ICM, so using them is optional.

Job Codes

Job codes correspond to the codes your company uses to identify positions within your organization. For example, CSR might be a code for Customer Service Representative. Each employee must be assigned a job code.

Job codes are commonly used to determine plan eligibility. For example, you can set up a plan rule that selects only employees with a job code of ACCEX. Job codes are also used in calculations. For example, a calculation formula might reference the salary or incentive values for a job code to generate earning amounts.

You can also set up rate groups and associate the records with job codes. If your compensation plan does not use the rate groups feature, then do not create this object. For more information about rate groups, see “Product Rates and Rate Groups” on page 91.

Locations

Locations describe the physical geographic addresses of Siebel ICM entities such as employees, organizations, customers, and channel partners. They are primarily used by transactions to identify where goods were shipped to or sold to, and to credit the sales accordingly. ICM sometimes uses locations in crediting, so using them is optional.

Payroll Systems

Payroll systems identify the external software systems used to pay employees. This information is necessary for establishing an interface with different payroll software systems. For example, payroll systems are used for employees paid by multiple systems, such as in merger or acquisition situations. Payroll systems are rarely employed in ICM, so using them is optional.

Salary Grades

Salary grades define pay ranges for job functions. A salary grade is expressed as a minimum salary and a maximum salary. You can also define a midpoint or average salary. Salary grades determine how earnings are calculated and applied.

Tasks for Setting Up Reference Data

This topic lists the tasks for setting up reference data. You can do these tasks in any order.

- “Setting Up Jobs” on page 80
- “Setting Up Salary Grades” on page 82
- “Setting Up Cost Centers” on page 83
- “Setting Up a Payroll System” on page 83
- “Setting Up Locations” on page 84
- “Setting Up Channel Segments” on page 84

Setting Up Jobs

To set up jobs, you must first add a job record. After the job has been added, you can add information to the record or edit the record at any time.

To add a job

1 Navigate to the Reference Data > Jobs view.

2 Click the New Job link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the job function.
Rate Group Code	If you are using rate groups, select a code. In conjunction with the product rate, the rate group identified by the selected code will determine the commission rate for this job for any given product. **NOTE:** A product can have multiple Rate Groups. To select the correct Product Rate Group for a participant, you can use the Rate Group defined for that participant's Employee, Job, or Organization record. For more information about product rates and rate groups, see "Product Rates and Rate Groups" on page 91.

4 In the Salary Amount form, complete the necessary fields.

NOTE: The salary range data can be referenced through calculation formulas and is not required for all plans.

Some fields are described in the following table.

Field	Comments
Minimum	Minimum salary an employee can earn.
Midpoint	Either the average or median salary an employee can earn.
Maximum	Maximum salary an employee can earn.

5 In the Incentive form, complete the necessary fields.

NOTE: The incentive range data can be referenced through calculation formulas and is not required for all plans.

Some fields are described in the following table.

Field	Comments
Threshold	Minimum performance level that must be attained in order to acquire earnings.

Field	Comments
Target	Target level of achievement for a period. This value is not connected to goals or credits, so it can be used independently of these values.
Maximum	Maximum amount that an employee can earn through incentives.

6 In the Payout form, complete the necessary fields.

NOTE: The earning range data can be referenced through calculation formulas and is not required for all plans.

Some fields are described in the following table.

Field	Comments
Maximum Upside	Maximum difference between salary and incentive earnings.
Variable Target %	Ideal percentage of an employee's total pay that comes from incentive earnings.
Guarantee %	Ideal percentage of an employee's total pay that comes from a guaranteed salary.

7 Click Save.

Setting Up Salary Grades

It is recommended that you set up salary grades in the first working period of your fiscal year because they are versioned entities.

To add a salary grade

1 Navigate to the Reference Data > Salary Grades view.

2 Click the New Salary Grade link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the salary grade.
Name	Name for the salary grade.

Field	Comments
Modifier	Can be used to represent aspects of a salary grade that is not specifically covered in another field. Can be any numeric value. **NOTE:** Modifiers are often referenced by calculation formulas and used in distribution rule conditions.
Minimum Amount	Minimum salary an employee can earn if assigned to the salary grade.
Midpoint Amount	Either the average or median salary.
Maximum Amount	Maximum salary an employee can earn if assigned.

4 Click Save.

Setting Up Cost Centers

To set up cost centers, first you must add a cost center record.

To add cost centers

1 Navigate to the Reference Data > Cost Centers view.

2 Click the New Cost Center link.

3 In the Basic Information form, complete the necessary fields.

4 Click Save.

Setting Up a Payroll System

To set up a payroll system, you must add a payroll system record.

To set up a payroll system

1 Navigate to the Reference Data > Payroll Systems view.

2 Click the New Payroll System link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the payroll system.
Contact	Name of the employee, contractor, or other person responsible for administering the payroll system.

4 Click Save.

Setting Up Locations

You must set up locations before entering or importing employee records.

To add a location

1 Navigate to the Reference Data > Locations view.

2 Click the New Location link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the location.
Name	Name of the employee, contractor, or other person responsible for administering this location.
Type	User-defined classification for a location.

4 Click Save.

Setting Up Channel Segments

To set up channel segments, you must add a channel segment record by performing the following procedure.

To add a channel segment

1 Navigate to the Reference Data > Channel Segments view.

2 Click the New Channel Segment link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the channel system.
Name	Name for the channel segment.

4 Click Save.

14 Defining Organizations and Organization Hierarchies

This chapter describes organizations and organization hierarchies. It includes the following topics:

- "About Organizations and Organization Hierarchies" on page 87
- "Process of Constructing an Organization Hierarchy" on page 88

About Organizations and Organization Hierarchies

Organizations generally represent the divisions and departments within a company.

If organization performance is necessary in determining participants' eligibility and earnings, organizations can have goals and credits assigned to them, just as participants do. For example, some bonus plans credit bonuses to employees if their department reaches a certain sales or performance level. In these plans, each employee's performance is added to the organization's overall performance. This overall performance determines the earning eligibility of all employees in the organization.

NOTE: While organizations can have goals and credits, organizations do not receive earnings. Only participants can receive earnings.

You can group organizations into an *organization hierarchy* to establish the reporting relationships between divisions or departments.

Organization hierarchies are primarily used by the system's rules. Plan rules, for example, may select participants based on which organization the participants report to. Distribution rules may generate credit records for a specific participant, for the organization that participant reports to directly, and for another organization higher up in the hierarchy. Hierarchies are also required if the performance records of lower-level organizations must be rolled up into higher-level organizations.

Only one organization hierarchy can be constructed for each operating unit. The organization at hierarchy level 1 reports to the operating unit itself.

Organizations and organization hierarchies are specific to the operating unit in which they are created. If multiple operating units exist in the system, the organizations and organization hierarchy of one operating unit cannot be shared with another operating unit. You can copy organizations and organization hierarchies from one operating unit to another, but the system treats the copied organizations as distinct and separate from the original organizations.

Process of Constructing an Organization Hierarchy

This topic describes the process of adding organizations and arranging them in an organization hierarchy. To construct an organization hierarchy, you perform the following tasks:

1. "Adding Organization Hierarchy Levels" on page 88
2. "Setting Up Organizations" on page 88
3. "Setting Up the Organization Hierarchy's Root Element" on page 89
4. "Adding Child Organizations to the Hierarchy" on page 90

Adding Organization Hierarchy Levels

Each organization is assigned to a hierarchy level when it is created. Thus, you must set up hierarchy levels before creating organizations. To set up an organization hierarchy level, follow this procedure.

This task is a step in "Process of Constructing an Organization Hierarchy" on page 88.

To add an organization hierarchy level

1. Navigate to the Organization > Organization Hierarchy Level Set view.
2. Click the Edit icon above the Levels list.
3. In the Add Level form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Order	Number that indicates this level's place in the hierarchy. For example, enter 3 to make this the third hierarchy level from the top.
Level Code	Code or text label that uniquely identifies or describes this hierarchy level.

4. Click the Add link.

Setting Up Organizations

To set up an organization, you must add an organization record by performing the following procedure.

This task is a step in "Process of Constructing an Organization Hierarchy" on page 88.

To add an organization

1 Navigate to the Organization > Organizations view.

2 Click the New Organization link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the organization.
Level Code	Identifying code for the hierarchy level at which the organization can be placed.
Location Code	Code that corresponds to where the organization is set up.
Status	Active indicates that this is the current organization.
Manager ID	Code of the employee that acts as the manager for the organization.
Modifier	Can be used to represent aspects of a salary grade that is not specifically covered in another field. Can be any numeric value. **NOTE:** Modifiers are often referenced by calculation formulas and used in distribution rule conditions.

4 In the Additional Information form, complete the necessary fields.

5 Click Save.

Setting Up the Organization Hierarchy's Root Element

After you have created organization records, you can construct an organization hierarchy. First, you assign one organization to be the top-most organization in the hierarchy. This is the *root element*.

NOTE: Only one organization can be assigned to hierarchy level 1. Organizations at all other hierarchy levels can have any number of other organizations assigned to them.

Set up a root element by performing the following procedure.

This task is a step in "Process of Constructing an Organization Hierarchy" on page 88.

To add the organization hierarchy's root element

1 Navigate to the Organization > Organization Hierarchy view.

2 Click the Create Hierarchy link.

3 Click the Add Root Element link.

4 To add the top-most organization in the hierarchy, search for the organization's code and select the organization.

Adding Child Organizations to the Hierarchy

After you have created the top-most organization in the hierarchy, you can add organizations as "children" of the top-most organization. These organizations can then be the "parents" of lower level "child" organizations.

To add child organizations, perform the following procedure.

This task is a step in "Process of Constructing an Organization Hierarchy" on page 88.

To add a child organization to the hierarchy

1 Navigate to Organization > Organization Hierarchy view.

2 On the Organization Hierarchy tree's title bar, click the Edit icon.

3 Find the parent organization that you want to add a child organization to, and then click its Add Node icon.

4 Enter search criteria that identify the organization you want to add, and then click Search.

5 In the Organizations Found list, locate the organization you want to add as a child and select its check box.

The organization tree reappears with the new child organization added to the parent organization you selected. Depending on the level of the child organization, you may have to open the hierarchy further to see it.

15 Defining Products and Product Hierarchies

This chapter describes products and includes the following topics:

- "About Products and Product Hierarchies" on page 91
- "Process of Constructing a Product Hierarchy" on page 92

About Products and Product Hierarchies

Products represent the categories of goods your company sells, or other kinds of offerings such as customer services or contracts. You can order products into a *product hierarchy*. Generic classifications of products occupy the highest levels of the hierarchy, with more specific classifications lower down and the most specific products at the lowest levels.

Product Measures and Measure Types

A *measure* is an object that tracks specific types or categories of performance. Measures are associated with every goal and credit record. They determine how those records are cumulated or rolled up through the organization.

You can associate one or more performance measures with each product or product category. Distribution rules use this association to determine the credited measure according to the product sold in a transaction.

A *measure type* is a category that groups measures together in some logical fashion. A measure type is referenced by a distribution rule. It allows the distribution rule to choose the correct measure, if multiple measures have been associated with products. Product measure types can apply to all products across all operating units.

Product measures and measure types are rarely used in ICM, and are optional.

Product Rates and Rate Groups

A *product rate* is 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*, an attribute that determines which product rate applies to a particular employee. Rate groups are assigned either directly to employees or indirectly through the employees' job codes. During plan calculation, product rates and rate groups determine what commission rate a participant should earn for the sale of a product.

In sales organizations, commission rates are sometimes determined by what product or type of product a participant has sold. For example, selling new equipment might carry a higher commission rate than selling replacement parts for those items. Additionally, the commission rate for a product sale may be further determined according to who sells it. Direct sales representatives, for example, may earn higher commission rates than inside sales representatives for sales of the same product. You can use rate groups to determine these different commission rates.

Process of Constructing a Product Hierarchy

This topic describes the process of adding products and arranging them in a product hierarchy. To construct a product hierarchy, you perform the following tasks:

1. “Setting Up Rate Groups” on page 92
2. “Adding Product Hierarchy Levels” on page 92
3. “Setting Up Products” on page 93
4. “Setting Up a Product Hierarchy’s Root Element” on page 95
5. “Adding Child Products to a Product Hierarchy” on page 95

Setting Up Rate Groups

You can add a rate group record to determine which product incentive rate will apply to a particular employee. To set up a rate group, follow this procedure.

This task is a step in “Process of Constructing a Product Hierarchy” on page 92.

To add rate groups

1. Navigate to the Reference Data > Rate Groups view.
2. Click the New Rate Group link.
3. In the Basic Information form, complete the necessary fields.
4. Click Save.

Adding Product Hierarchy Levels

Each product is assigned a hierarchy level when it is created. Thus, you must set up hierarchy levels before creating products. To set up a product hierarchy level, follow this procedure.

This task is a step in “Process of Constructing a Product Hierarchy” on page 92.

To add a product hierarchy level

1. Navigate to the Reference Data > Product Hierarchy Level Set view.
2. Click the Edit icon above the Levels list.

3 In the Add Level form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Order	Number that indicates this level's place in the hierarchy. For example, enter 3 to make this the third hierarchy level from the top.
Level Code	Code or text label that uniquely identifies or describes this hierarchy level.

4 Click the Add link.

Setting Up Products

To set up products, you must first add a product record. You can then associate rate groups and measure types with the product.

To associate a measure type with a product, you must first create measures. For instructions, see the chapter on measures in *Siebel Incentive Compensation Management Administration Guide*.

This task is a step in "Process of Constructing a Product Hierarchy" on page 92.

To add a product

1 Navigate to the Reference Data > Products view.

2 Click the New Product link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the product.
Level Code	Code that corresponds to the product's level assigned in the product hierarchy.

4 Click Save.

5 In the Rates list, click the Edit icon.

6 In the Default Rate field, enter the base or default commission rate that participants may earn on sales of this product.

This is the commission rate that will be used if a calculation formula needs to find a product's commission rate but it cannot find the appropriate rate group for a participant.

To associate a rate group with a product

1 Navigate to the View Product page.

2 Click the Add Rate Group Rate link.

The Measures section associates performance measures with the product to create distribution rule credits.

3 In the Rate Group Rate form, complete the necessary fields.

For complete details on setting up rate groups and product rates, see "Product Rates and Rate Groups" on page 91.

4 Click Add.

To associate a measure type with a product

1 Navigate to the View Product page.

2 Click the Add Measure Type link.

The Measures section associates performance measures with the product to allow the system to calculate distribution rule credits. See "Example of Product Measures and Measure Types" on page 94.

3 In the Measures section, do one of the following:

- Select New Type to add a new product measure type, and then enter the new type.
- Select Existing Type and select one of the product measure types that have already been created for other products.

4 If you selected New Type, select a Measure Code to associate with the new product measure type.

5 Click Add.

Example of Product Measures and Measure Types

When setting up measure associations with a product, you select a Product Measure Type before selecting a specific measure. Different products can have different measures assigned to them using the same Product Measure Type, as shown in Table 8.

Table 8. Product Measures and Measure Types

Product	Measure	Measure Type
Hourly Consulting	CSG	Type1
Enterprise Software	Quota	Type1
Computer Monitor	Non-Quota	Type1

In this example, when you configure the distribution rule credit, you set the Distribution Measure to the Product's Measure and the Measure Type to Type1. When the crediting service creates the distribution, the product on the transaction line determines which measure will be used.

Setting Up a Product Hierarchy's Root Element

You can construct the product hierarchy when you create products. You create the hierarchy by assigning one product to be the top-most product in the hierarchy, then adding more products as children or subcategories. These products can then be the parents of lower-level child products. The products at the lowest level represent the most specific products, while higher-level products denote categories and subcategories of products.

To set up the product hierarchy, you must first add a root element. The root element is the top-most organization or product in a hierarchy to which all other organizations or products report.

This task is a step in "Process of Constructing a Product Hierarchy" on page 92.

To add root elements

1. Navigate to the Reference Data > Product Hierarchy view.
2. Click the Create Hierarchy link.
3. Click the Add Root Element link.
4. Enter search criteria to identify the product you want to designate as the root element, and then click Search.
5. In the Products Found list, locate the item you want to designate as the root element, and then select its check box.

   The Product Hierarchy page reappears with the product you selected as the root element.

Adding Child Products to a Product Hierarchy

You can construct the product hierarchy when you create products. You create the hierarchy by assigning one product to be the top-most product in the hierarchy, then adding more products as children or subcategories. These products can then be the parents of lower-level child products. The products at the lowest level represent the most specific products, while higher-level products denote categories and subcategories of products.

You can add child products and then associate them to root elements.

This task is a step in "Process of Constructing a Product Hierarchy" on page 92.

To add child products

1. Navigate to the Reference Data > Product Hierarchy view.
2. Click the Edit icon.
3. In the hierarchy tree, find the parent product to which you want to add a child product, and then click the Add Node icon.
4. Enter search criteria to identify the product you want to add as a child, and then click Search.

5 In the Products Found list, locate the product you want to add as a child, and then select its check box.

The hierarchy tree reappears with the new child product added to the parent product you selected. Depending on the level of the child product, you may have to open the hierarchy further to see it.

16 Setting Up Participants

This chapter describes setting up participants. It includes the following topics:

- "About Participants" on page 97
- "Setting Up Employees" on page 98
- "Process of Changing Employee Job Status" on page 101
- "Process of Setting Up Channel Partners" on page 103
- "Process of Setting Up Customers" on page 106

About Participants

Participants are the individuals, businesses, and other entities that receive earnings from an incentive plan. There are three types of participants—employees, channel partners, and customers.

An incentive plan can calculate earnings for only one type of participant. If an entity is treated as more than one participant type, such as a business that acts as a channel partner and is also a customer, then you must set that entity up separately for each participant type. Earnings under each plan are not consolidated. This means, for example, that earnings under a channel partner incentive plan cannot be reported together with earnings under a customer incentive plan.

Employees

Employees include members of your company that have some or all of their pay determined by incentive plans. Employees who participate in one or more incentive plans must be entered in the Siebel ICM database.

For most Siebel ICM implementations, employee data is imported from an external system, such as a Human Resources Management program. You can also enter employees and edit their data directly in ICM.

Channel Partners

Channel partners include any business or third-party entity with which the company has a partnership and through which the company sells products or services. Incentive plans typically reward channel partners for promoting and selling your products or for bringing in new customers.

In most ICM implementations, channel partner data is imported from an external database or channel partner software system. You can also add channel partners and edit their data directly in ICM.

Customers

Customers are the people or businesses who buy your company's products or services. Typically, distribution rules reference customers as selection criteria for transactions. This allows salespeople to acquire earnings based on the customer account to which a product or service was sold.

Customers can also be participants. Incentive plans reward customers based on their purchase history or loyalty to particular products.

Setting Up Employees

To set up employees, you must add an employee record. To add an employee record, follow these procedures.

NOTE: Before employees can be set up or imported into the system, it is recommended that you set up reference data, products, and organizations. If any of these items have not been set up yet, you can still create employee records. When these items have been set up, you can return to the employee records and enter the missing information as appropriate.

To add an employee record

1 Navigate to the Organization > Employees view.

2 Click the New Employee link, and complete the necessary fields. Some fields are described in the following table.

Field	Comments
Employee ID	Unique identifying code for the employee.
Version Start Period	Versioning information, automatically completed by the system. Shows the calendar period in which this version of the employee record was created or modified.
Version Expiration Period	Versioning information, automatically completed by the system. Shows the last calendar period in which this version of the employee record is current. For example, when you create and save an employee record, this field displays None. However, if you modify an existing employee record's Home Address field in FY2006, Period 3, change the period to FY2006, Period 1, and then return to that same employee record, this field displays FY2006, Period 2.

3 Click Next.

To add details about an employee

1 In the Company Directory form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Location Code	Code that corresponds to where the employee works.
Language	Preferred language of the employee.

2 If this employee is also an ICM user, set up the employee's user profile by completing the necessary fields in the User Details form.

Some fields are described in the following table.

Field	Comments
User Name	Employee's unique user name. **NOTE:** The user name may be different from the employee's name or code.
Role	Determines what pages and functions the user will be allowed to access.

3 In the Hire Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Pay Eligible	Select to make the employee eligible to receive earnings. To participate in a calculation, summary or payment plan, an employee must be pay eligible. If cleared, the employee will still receive credits. Earnings will also still be computed, but the system will not generate summarized earnings for this employee.
Payroll System Code	Used in distribution rule or plan rule selection formulas. Available currencies are determined by the operating unit administrator when the Operating Unit is created.

Field	Comments
Currency Code	The participant will be paid in the selected currency. If different than the operating unit currency, the currency conversion must be defined.
Rate Group Code	If you are using rate groups, select a code. In conjunction with the product rate, the rate group identified by the selected code determines the commission rate for this job for any given product. If you want the rate group associated with the employee's job code to apply to this employee, then leave this field set to None. **NOTE:** A product can have multiple Rate Groups. To select the correct Product Rate Group for a participant, you can use the Rate Group defined for that participant's Employee, Job, or Organization record. This field might, for example, be used by a smart attribute in Crediting to determine which product rate group to use for this employee. For more information about product rates and rate groups, see "Product Rates and Rate Groups" on page 91.

4 Click Next.

To add job information for employees

1 In the Job Status form, complete the necessary fields. Some fields are described in Table 9.

Table 9. Job Status Fields

Field	Comments
Job Code	Job function for the employee.
Status	*Active* indicates that this is the employee's current, active job function.
Start Date	Date when the employee first took on this job function.
Exempt	Select if the employee is in a non-hourly wage employee.
Supervisor ID	If the employee has a supervisor, enter the supervisor's employee code.
Organization Code	Organization to which this employee reports directly.
Cost Center Code	Cost center that this employees' variable pay is charged against.

2 In the Salary form, complete the necessary fields. Some fields are described in Table 10.

Table 10. Salary Fields

Field	Comments
Salary Per Period	Employee's salary per pay period. Informational only.
Salary Grade Code	What salary grade the employee is in. Can be used in Crediting or in any of the plan services.

3 In the Additional Information form, complete the necessary fields.

4 Click Done.

Process of Changing Employee Job Status

This topic describes the process for keeping employee job records current. To maintain employee participant job records, you perform the following tasks:

- "Modifying Employee Job Information" on page 101
- "Changing an Employee's Job" on page 102

Modifying Employee Job Information

To modify job information for an employee without changing that employee's job, follow this procedure.

To modify employee job information

1 Navigate to the Organization > Employees view.

2 In the Employees Found list, locate the employee whose job information you want to modify and click its View icon.

3 In the Job Information section, click the Edit icon.

4 In the employee job display, do the following:

   a In the Job Status form, make any necessary changes to the field values. Some fields are described in Table 9 on page 100.

b In the Salary form, make any necessary changes to the field values. Some fields are described in Table 10 on page 101.

NOTE: The employee job record is a subrecord of the employee record. The field values in the Version Information form apply to the employee record, *not* the employee job subrecord. Therefore, the Version Information values do not change when you modify the employee job information.

5 Click Save.

Changing an Employee's Job

When an employee switches positions or leaves the company, you inactivate the employee's old job record. When an employee switches positions, you then add a new job history record to the employee record.

This task is a step in "Process of Changing Employee Job Status" on page 101.

To inactivate an employee's job status

1 Navigate to the Organization > Employees view.

2 In the Employees Found list, locate the employee whose old job you want to inactivate and click its View icon.

3 In the Job Information section, click the Edit icon.

4 In the Status field, enter Inactive.

5 In the End Date field, enter the last day the employee was on the old job.

6 Click Save.

To add a new job record to an employee

1 Navigate to the Organization > Employees view.

2 In the Employees Found list, locate the employee whose new job you want to add and click its View icon.

3 In the Job Information section, click the New Job icon.

4 In the Job Status form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Job Code	Employee's new job function.
Start Date	Date when the employee first took on this job function.
Status	*Active* indicates that this is the employee's current, active job function.

Field	Comments
Supervisor ID	If the employee has a supervisor, enter the supervisor's employee code.
Organization Code	The organization to which the employee reports directly. Depending on the rate group, you must do one of the following: ■ If you are using rate groups in conjunction with product rates to determine commission rates, select the appropriate Rate Group for the employee. For more information on rate groups, see "Product Rates and Rate Groups" on page 91. ■ If you want the rate group of the employee's job code or organization to be used for this employee, then leave this field blank.

5 In the Salary form, complete the necessary fields

Some fields are described in Table 10 on page 101.

6 In the Additional Information form, complete the necessary fields.

7 Click Save.

Process of Setting Up Channel Partners

This topic describes the process of setting up channel partner participants in ICM. To set up channel partner participants, you perform the following tasks.

NOTE: Before you set up or import channel partners, employee records must already be set up. This allows you to assign employees as channel managers for the channel partner accounts. See "Setting Up Employees" on page 98.

1 "Adding Channel Partners" on page 103

2 "Adding Channel Partner Details" on page 104

3 "Adding Channel Partner Contacts" on page 105

4 "Adding Channel Partner Certifications" on page 105

Adding Channel Partners

To set up a channel partner, you must first add a channel partner record. To add a channel partner record, follow this procedure.

This task is a step in "Process of Setting Up Channel Partners" on page 103.

To add a channel partner

1 Navigate to the Organization > Channel Partners view.

2 Click the New Channel Partner link.

3 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Channel Partner ID	Unique identifying code for the channel partner.
Channel Partner Name	Unique name for the channel partner.
Tax ID	Federal tax ID number.

4 Click Save.

Adding Channel Partner Details

To add details to a channel partner record, follow this procedure.

This task is a step in "Process of Setting Up Channel Partners" on page 103.

To add details about channel partners

1 Navigate to the Organization > Channel Partners view.

2 In the Channel Partners Found list, locate the channel partner whose details you want to add and click its View icon.

3 In the Attributes section, click the Edit icon.

4 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Channel Manager ID	Code for the employee who acts as the internal manager for this channel partner.
Commisionable	Select to make the employee eligible to receive earnings. To participate in a calculation, summary or payment plan, an employee must be commissionable. If cleared, the employee will still receive credits. Earnings will also still be computed, but the system will not generate summarized earnings for this employee.
Parent Channel Partner ID	If this partner is a branch of a national or regional channel partner, enter the Parent Channel Partner ID.
Participant Currency	Currency in which payments should be made to the channel partner.

Field	Comments
Partner Rank	Can be used as a way of selecting partners for plan eligibility.
Tier	Can be used as selection criteria for plan or distribution rule eligibility.

5 Click Save.

Adding Channel Partner Contacts

To add contact information to a channel partner record, follow this procedure.

This task is a step in "Process of Setting Up Channel Partners" on page 103.

To add a contact for a channel partner

1 Navigate to the Organization > Channel Partners view.

2 In the Channel Partners Found list, locate the channel partner to which you want to add a contact and click its View icon.

3 In the Contacts section, click the Add Contact icon.

4 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Contact ID	Unique identifying code for the contact.
Primary Contact	Only one contact may be the primary contact. If another contact is already the primary contact, selecting this check box for the current contact clears the check box for the other contact.

5 Click Save.

Adding Channel Partner Certifications

To add a certification to a channel partner record, follow this procedure.

This task is a step in "Process of Setting Up Channel Partners" on page 103.

To add a certification to a channel partner

1 Navigate to the Organization > Channel Partners view.

2 In the Channel Partners Found list, locate the channel partner to which you want to add a certification and click its View icon.

3 In the Certifications section, click the Add Certification icon.

4 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Certification Code	Type of certification the partner has earned. Can be used if your company sells products or services that require certification for a partner to be a qualified user or reseller.
Level of Certification	Channel partner's level of certification.
Number of Certified Users	Number of users at this level of certification.

5 Click Save.

Process of Setting Up Customers

This topic describes the process of setting up customer participants in ICM. To set up customer participants, you perform the following tasks.

NOTE: Before you set up or import customers, employee records must already be set up. This allows you to assign employees as account managers or national account representatives for the customer accounts. See "Setting Up Employees" on page 98.

1 "Adding Customer Records" on page 106

2 "Adding Customer Details" on page 107

3 "Adding Customer Locations" on page 108

4 "Adding Customer Contacts" on page 109

Adding Customer Records

To set up a customer, you must first add a customer record. To add a customer record, follow this procedure.

This task is a step in "Process of Setting Up Customers" on page 106.

To add a customer

1 Navigate to the Organization > Customers view.

2 Click the New Customer link.

3 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Customer ID	Unique identifying code for the customer.
Company and Individual	Both types of customers (that is, company and individual) are treated the same way. Selecting Company or Individual determines which data fields appear on the next page for recording customer identity data.

4 In the Address Information form, complete the necessary fields.

5 Click Save.

Adding Customer Details

To add details to a customer record, follow this procedure.

This task is a step in "Process of Setting Up Customers" on page 106.

To add details about a customer

1 Navigate to the Organization > Customers view.

2 In the Customers Found list, locate the customer whose details you want to add and click its View icon.

3 In the Details section, click the Edit icon.

4 Depending on whether the customer is an individual or a company, do one of the following:

- **Individual.** In the Individual Information form, complete the necessary fields.
- **Company Information.** In the Company Information form, complete the necessary fields.

The remaining fields on this page are the same whether the customer is an individual or a company.

5 In the Attributes form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Commisionable	Select to make the customer eligible to receive earnings. To participate in a calculation, summary or payment plan, a customer must be commissionable. If cleared, the customer will still receive credits. Earnings will also still be computed, but the system will not generate summarized earnings for this customer.
Customer Type	Can be used in distribution rule and plan rule selection criteria to indicate how favored a customer is. **NOTE:** A customer's level is *not* related to the levels assigned to organizations, and customers are not part of a hierarchy.
Channel Segment Code	Channel through which sales are made to the customer—for example, Direct Sales or VAR Sales.
Rank	Can be used as a way of selecting customers for plan eligibility.
Good Credit Standing	Indicates that the customer has consistently paid its bills on time. Can be used to determine which customers are eligible for incentive plan participation.
Currency Code	The currency in which the customer does business or in which the customer should be paid.
Parent Customer ID field	If this customer is a local branch of a national or regional customer account, enter that customer's code.
Account Manager ID	Unique code of the employee that acts as the internal account manager for this customer.
National Account Rep ID	Indicates which employee acts as the representative for the corresponding national account. **NOTE:** This may be the same as the Account Manager ID.
Lead Source	Indicates whether the lead that brought in this customer's business was a Channel Partner or another Customer.
Lead Source ID	If a lead type is selected, enter that lead's code—that is, customer or channel partner.

6 Click Save.

Adding Customer Locations

To add locations to a customer record, follow this procedure.

This task is a step in "Process of Setting Up Customers" on page 106.

To add locations for the customer

1. Navigate to the Organization > Customers view.
2. In the Customers Found list, locate the customer whose locations you want to add and click its View icon.
3. In the Location section, click the Add Location icon.
4. In the Location form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Location ID	Unique identifying code for the location.
Business Purpose	Business purpose that the location is used for.

5. Click Save.

Adding Customer Contacts

To add contacts to a customer record, follow this procedure.

This task is a step in "Process of Setting Up Customers" on page 106.

To add a customer contact

1. Navigate to the Organization > Customers view.
2. In the Customers Found list, locate the customer whose contacts you want to add and click its View icon.
3. In the Contacts section, click the Add Contact icon.
4. Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Contact ID	Unique identifying code for the contact.
Primary Contact	Only one contact may be the primary contact. If another contact name is already the primary contact, selecting this check box for the current contact clears the check box for the other contact.

5. Click Save.

17 Defining Territories and Regions

Compensation administrators and sales managers set up and manage territories. This chapter describes territories in Siebel ICM and includes the following topics:

- "About Territories and Regions" on page 111
- "Process of Constructing a Region and Territory Hierarchy" on page 112

About Territories and Regions

Territories represent the localities in which a company's sales division operates. Territories are used in conjunction with distribution rules to determine how transaction credits are distributed among the sales division's employees. The system identifies the applicable territory based on location data from the transaction—for example, city, state, or ZIP Code.

Territories are at the bottom of a hierarchy. Territories report to *regions*, which can encompass either no territories, one territory, or multiple territories. Regions report to higher-level regions, and ultimately to a single region that represents the company's entire sales area.

Employees are assigned to territories. Multiple employees can be assigned to a single territory, and an employee can be assigned to multiple territories. Territories have managers, and territories can receive credits from transactions.

Territory Qualifiers

A major part of a territory's definition is its *qualifiers*. Qualifiers determine whether or not specific transactions qualify for that territory. When a transaction does qualify for a territory, the system generates credits for the territory and for employees who share that territory.

A qualifier consists of one or more *conditions*, which test certain data fields on a transaction against defined criteria. If a transaction matches all the conditions in a qualifier, then the transaction passes that qualifier. If a transaction fails one condition, then it fails the qualifier. If multiple qualifiers exist for a territory, then the transaction has to match only one qualifier. Thus, a transaction may fail to meet conditions in one qualifier but may meet all the conditions in another qualifier, and would then qualify for the territory.

After a transaction has qualified for one or more territories, a distribution rule uses those territories to distribute credits for the transaction. Credits can be given directly to the territories, to employees who are assigned to the territories, to territory managers, or to any combination of these. How credits are distributed is determined by the recipient templates chosen for the distribution rule.

Regions do not have qualifiers. Thus, a transaction can only qualify for a territory. However, regions can receive credits if credits are rolled up from the territories to the regions.

Process of Constructing a Region and Territory Hierarchy

This topic describes the process of adding regions and territories, and arranging them in a hierarchy. To construct a region and territory hierarchy, you perform the following tasks:

1. "Adding Region Hierarchy Levels" on page 112
2. Setting up regions:
   a. "Adding Regions" on page 113
   b. "Adding Employees to a Region" on page 113
   c. "Adding Channel Partners to a Region" on page 114
3. Setting up territories:
   a. "Adding Territories" on page 115
   b. "Adding Qualifiers to a Territory" on page 115
   c. "Adding Employees to a Territory" on page 116
   d. "Adding Channel Partners to a Territory" on page 116
4. "Setting Up the Region and Territory Hierarchy's Root Element" on page 117
5. "Adding Child Regions and Territories to the Hierarchy" on page 117

Adding Region Hierarchy Levels

Each region is assigned to a hierarchy level when it is created. Thus, you must set up hierarchy levels before creating regions. To set up a region hierarchy level, follow this procedure.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add a region hierarchy level

1. Navigate to the Organization > Territory Hierarchy Level Set view.
2. Click the Edit icon above the Levels list.
3. In the Add Level form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Order	Number that indicates this level's place in the hierarchy. For example, enter 3 to make this the third hierarchy level from the top.
Level Code	Code or text label that uniquely identifies or describes this hierarchy level.

4 Click the Add link.

Adding Regions

To set up a regions, you must first add a region record.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add a region

1 Navigate to the Organization > Regions view.

2 Click the New Region link.

3 In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the region.
Channel Segment Code	Channel through which sales are made in the region. Associating a channel segment with a region tracks the channels through which a company sells its products in that region. You can configure your plan to use this field value in Crediting, or any of the plan services.
Level Code	Code that corresponds to the hierarchy level assigned to the region.
Manager ID	Code of the employee who acts as the manager for the region.

4 Click Save.

Adding Employees to a Region

After a region record has been created, you can add employees to the region.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add an employee to a region

1 Navigate to the Organization > Regions view.

2 Identify the region to which you want to add an employee and click its View icon.

3 In the Employees list, click the Add Employee icon.

4 In the Employee form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Role in Region	Employee's position in this region.
Start Date	Date when the employee became part of this region. **NOTE:** Regardless of the date entered in this field, the Start Period for the employee will be the working period in which the employee is added to the region.
End Date	If known, date the employee will no longer be part of the region.

5 Click Add.

Adding Channel Partners to a Region

After a region record has been created, you can add channel partners to the region.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add a channel partner to a region

1 Navigate to the Organization > Regions view.

2 Identify the region to which you want to add a channel partner and click its View icon.

3 In the Channel Partner section, click the Add Channel Partner icon.

4 In the Channel Partner form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Role in Region	Channel partner's position in this region.
Start Date	Date when the channel partner became part of this region. **NOTE:** Regardless of the date entered in this field, the Start Period for the channel partner is the working period in which the channel partner is added to the region.
End Date	If known, date that the channel partner will no longer be part of the region.

5 Click Add.

Adding Territories

To set up territories, you must first add a territory record.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add a territory

1. Navigate to Organization > Territories view.
2. Click the New Territory link.
3. In the Basic Information form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Code	Unique identifying code for the territory.
Channel Segment Code	Channel through which sales are made in the territory. Associating a channel segment with a territory tracks the channels through which a company sells its products in that territory. You can configure your plan to use this field value in Crediting, or any of the plan services.
Manager ID	Code of the employee who acts as the manager for the territory.

4. Click Save.

Adding Qualifiers to a Territory

After creating a territory record, you can add qualifiers to the territory.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add qualifiers to the territory

1. Navigate to the Organization > Territories view.
2. Identify the territory to which you want to add a qualifier and click its View icon.
3. In the Qualifiers section, click the Add Qualifier icon.
4. In the Conditions section, click the Add Condition icon.

   The Territory Qualifier dialog box appears.

5 In the Territory Qualifier Condition form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Attribute	Transaction attribute.
Operator	Comparison operator.
Value	Value against which to test the attribute.

6 Click Add.

7 Repeat Step 4 through Step 6 for each condition you want to add to the qualifier.

Adding Employees to a Territory

After creating a territory record, you can add employees to the territory.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add employees to the territory

1 Navigate to the Organization > Territories view.

2 Identify the territory to which you want to add an employee and click its View icon.

3 In the Employees section, click the Add Employee icon.

4 In the Employee form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Role in Territory	Employee's position in this territory.
Start Date	Date when the employee became part of this territory. **NOTE:** Regardless of the date entered in this field, the effective start period for the employee in this territory is the working period in which the employee is added to the territory.
End Date	If known, date that the employee will no longer be part of the territory.

5 Click Add.

Adding Channel Partners to a Territory

After creating a territory record, you can add channel partners to the territory.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add channel partners to the territory

1 Navigate to the Organization > Territories view.

2 Identify the territory to which you want to add a channel partner and click its View icon.

3 In the Channel Partners section, click the Add Channel Partner icon.

4 In the Channel Partners form, complete the necessary fields. Some fields are described in the following table.

Field	Comments
Role in Territory	Channel partner's position in this territory.
Start Date	Date when the channel partner became part of this territory. **NOTE:** Regardless of the date entered in this field, the Start Period for the channel in the territory is the working period in which the channel is added to the territory.
End Date	If known, date that the channel partner will no longer be part of the territory.

5 Click Add.

Setting Up the Region and Territory Hierarchy's Root Element

To set up the root element, or top level to the hierarchy, follow this procedure.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To set up the root element of the region and territory hierarchy

1 Navigate to the Organization > Territory/Region Hierarchy view.

2 Click the Create Hierarchy link.

3 Click the Add Root Element link.

NOTE: Only regions appear in the list of available entities for the root element, because only regions can be parents to other regions and territories.

4 In the Regions Found list, identify the region you want to designate as the root element, and select its check box.

Adding Child Regions and Territories to the Hierarchy

To add child regions or territories to a parent region, follow this procedure.

This task is a step in "Process of Constructing a Region and Territory Hierarchy" on page 112.

To add child regions or territories

1. Navigate to Organization > Territory/Region Hierarchy view.
2. Click the Edit icon.
3. In the hierarchy tree, find the parent region to which you want to add a child region or territory, and click its Add Node icon.

   NOTE: Both regions and territories appear in the list of available entities, because both regions and territories can be children of regions. Territories do not have a value in the Level Order field because they are always the lowest level of the hierarchy.

4. In the Territories and Regions Found list, locate the item you want to add as a child and select its check box.

   The hierarchy tree reappears with the new child region or territory added to the parent region you selected. Depending on the level of the child region or territory, you may have to open the hierarchy further to see it.

18 Establishing Password Policy

This chapter describes password policy alternatives in ICM and includes the following topics:

- "About ICM Password Policy" on page 119
- "Setting the ICM Password Policy" on page 120
- "About Configurable Password Properties" on page 120
- "Filtering Security-Related Log Messages" on page 125

About ICM Password Policy

Configurable password policy options give you additional flexibility to enforce your company's password security standards. ICM has a native password policy, a non-strict password policy, and a configurable password policy.

Native Password Policy

The native password policy is the default password policy that comes with your ICM application. None of its properties are configurable. This policy enforces the following standards:

- Hash Method of MD5
- Password Minimum Length (6 characters)
- Password Strict Syntax

  Strict syntax means that the password contains both numbers and letters, but no spaces.

Non-Strict Native Password Policy

ICM can use a non-strict version of the native password policy. None of its properties are configurable. The policy enforces the following standards:

- Hash Method of MD5
- Password Minimum Length (6 characters)

Configurable Password Policy

ICM can use a configurable password policy. All of its properties are configurable by editing the "password.policy" properties in the SiebelICMConfig.xml.in file. It requires the use of the CastorLoginModule. The policy enforces the following standards:

- Failed Attempt Limit
- Number of Passwords to Remember

- Failed Attempt Timeframe
- Hash Method
- Lock on Failed Attempt Limit
- Lock Timeout
- Password Minimum Age
- Password Minimum Length
- Password Strict Syntax
- Safe Modification

Setting the ICM Password Policy

The process of setting up ICM security consists of deciding which of ICM's available password policy options to use and then performing the following task.

To set the ICM password policy

1. Navigate to the <STAGING>/etc/JConfig/ directory.
2. In a text editor or XML editor, open the SiebelICMConfig.xml.in file.
3. In the SiebelICMConfig.xml.in file, change the value of the `password.policy.factory` property to one of the values in the following table, according to which password policy you want to set.

Password Policy	Property Value	Then Go To
Configurable	`com.motiva.ce.ds.password.ConfigurablePasswordPolicyFactory`	Step 4
Native	`com.motiva.ce.ds.password.NativePasswordPolicyFactory`	Step 5
Non-strict native	`com.motiva.ce.ds.password.NonStrictNativePasswordPolicyFactory`	Step 5

4. For configurable password policy only, in the SiebelICMConfig.xml.in file, change the values of the other properties as needed to match your company's password policy standards.

   For information about these properties and their valid values, see "About Configurable Password Properties" on page 120.
5. Save and close the SiebelICMConfig.xml.in file.
6. Navigate to the <STAGING>/deploy directory and run the following command:

   ```
   ant deploy-config
   ```
7. Restart the application server.

About Configurable Password Properties

The configurable password properties in the SiebelICMConfig.xml.in file are as follows:

- "Hash Method" on page 121
- "Failed Attempt Limit" on page 121
- "Failed Attempt Timeframe" on page 122

- "Lock on Failed Attempt Limit" on page 122
- "Lock Timeout" on page 122
- "Password Minimum Age" on page 123
- "Password Minimum Length" on page 123
- "Password Strict Syntax" on page 123
- "Safe Modification" on page 124
- "Number of Passwords to Remember" on page 124

Hash Method

The algorithm that specifies how ICM stores and checks passwords. Using a one-way hash to encrypt passwords prevents users with access to the RDMBS from using other people's passwords to gain access to ICM.

Property

`password.policy.hashMethod`

Valid Values

Value	Comments
`MD5`	The MD5 message digest algorithm as defined in RFC 1321.
`SHA`	SHA-1, the Secure Hash Algorithm as defined in Secure Hash Standard, NIST FIPS 180-1.

Default

`MD5`

Failed Attempt Limit

The number of failed attempts allowed before the system locks a user out.

Property

`password.policy.failedAttemptLimit`

Valid Values

Any positive integer. Setting `password.policy.lockOnFailedAttempt` to `false` causes ICM to ignore this check.

Default
10

Failed Attempt Timeframe

The time (in minutes) during which the failed attempts must occur.

Property
password.policy.failedAttemptTimeframeMinutes

Valid Values
Any positive integer. Setting password.policy.lockOnFailedAttempt to false causes ICM to ignore this check.

Default
2

Lock on Failed Attempt Limit

Determines whether the system locks a user out after a number of failed attempts.

Property
password.policy.lockOnFailedAttempt

Valid Values
true
false

Default
true

Lock Timeout

If a user is locked out, the length of time in minutes that user remains locked out.

Property
password.policy.lockTimeoutMinutes

Valid Values
Any positive integer.

Default
60

Password Minimum Age

Length of time in minutes until a user can change her password.

Property
password.policy.passwordMinimumAgeMinutes

Valid Values
Any positive integer. A value of -1 causes ICM to ignore this setting.

Default
60

Password Minimum Length

Sets the minimum length in number of characters for a password.

Property
password.policy.passwordMinimumLength

Valid Values
Any positive integer. A value of -1 causes ICM to ignore this setting.

Default
6

Password Strict Syntax

Determines whether to enforce strict syntax; that is, whether to make sure the password contains both numbers and letters but no spaces.

Property
password.policy.strictSyntax

Valid Values
true
false

Default
true

Safe Modification

Requires a user to supply the old password when changing to a new password. *Not* configurable.

Property
`password.policy.safeModification`

Valid Value
`true`

Default
`true`

Number of Passwords to Remember

Prevents repeated use of the same password over a specified period of time. Using this property with the Password Minimum Age property allows you to prevent a user from changing his password when it expires and immediately changing it back to its former value.

Property
`password.policy.numberOfPasswordsToRemember`

Valid Values
Any positive integer. A value of `-1` causes ICM to ignore this setting.

Default
3

Filtering Security-Related Log Messages

For audit purposes, ICM can record security-related messages in its audit log file, Siebel_ Audit_Log.txt. These messages can relate to user creation, password violations, password changes, password expirations, and so on.

For more information about ICM's error-logging functions, see Chapter 27, "Setting Up Debugging and Logging."

To filter the security-related messages that ICM records in its audit log, follow this procedure.

To filter security-related log messages

1. Determine your system's logging mode by doing the following:
   a. Navigate to the <STAGING>/deploy/ directory.
   b. In a text editor, open the deploy.default.properties file.
   c. Check the value for the deploy.logging.mode property.

      This value should be either debug or release.
2. In the <STAGING>/config/ directory, depending on the logging mode your system is using, open either the log4j.debug.in file or the log4j.release.in file.

   TIP: If you want the log changes to occur regardless of ICM's logging mode, you can make the changes described in the following steps in both the log4j.debug.in and log4j.release.in files.
3. Locate the category that routes Audit Log messages:

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

   This records all Audit-category messages with a priority of info or higher to the Audit Log.
4. To filter the messages that ICM logs, replace this category with a category for each message type you want the system to log, with the following syntax for each category name:

```
<category name="Audit.MESSAGE_TYPE">
```

Replace MESSAGE_TYPE with the specific message type you want logged. Security-related message types are described in the table that follows.

Message Type	Logged When...	Message Example
NEW_ACCOUNT	A user account is created.	[2005-09-16 15:34:00 759] (HttpProcessor[8080][3]:) [FATAL] [Audit.NEW_ACCOUNT] Account for [blinkou3] has been created.
NEW_PASSWORD	A password is changed.	[2005-09-16 15:57:19 077] (HttpProcessor[8080][4]:) [FATAL] [Audit.NEW_PASSWORD] Account for [blinkou3] has new password.
FAILED_LOGIN	A user fails authentication.	[2005-09-16 15:40:53 741] (HttpProcessor[8080][3]:) [FATAL] [Audit.FAILED_LOGIN] Login attempt by [blinkou3] failed.
LOCKED_ACCOUNT	A user account is locked.[1]	[2005-09-16 15:41:25 367] (HttpProcessor[8080][4]:) [FATAL] [Audit.LOCKED_ACCOUNT] Exceeded failed attempt limit. Account for [blinkou3] has been locked.
PASSWORD_EXPIRING	A user's password is about to expire.	[2005-09-16 15:45:25 335] (HttpProcessor[8080][4]:) [FATAL] [Audit.PASSWORD_EXPIRING] Password due to expire. Account for [blinkou3] has [3] more logins before lockout.

1. Lockout can occur if a user exceeds the failed attempt limit, or if her password has expired.

For example, to record only FAILED_LOGIN messages in the log, replace the Audit category with the following code lines:

```
<!-- Audit log categories -->
<category name="Audit.FAILED_LOGIN">
       <priority value="info" />
       <appender-ref ref="AUDIT" />
</category>
```

As another example, to record FAILED_LOGIN, NEW_PASSWORD, and NEW_ACCOUNT messages in the log, replace the Audit category with the following code lines:

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

5 Save and close the log4j file.

6 Apply your changes by doing the following:

a Navigate to the <STAGING>/deploy/ directory.

b Run the following command:

```
ant deploy-logging
```

c Restart the application server.

NOTE: ICM uses Jakarta log4j to control logging. log4j provides a wide variety of logging capabilities for ICM. For more information about configuring your logging, see the log4j documentation.

19 Setting Up the Publish Service

This chapter describes how to configure and use the Publish Service. This chapter contains the following topics:

- "About the Publish Service" on page 129
- "Process of Setting Up and Running the Publish Service" on page 129

NOTE: The Publish Service is available only in Siebel ICM version 7.8.2.2 or higher.

About the Publish Service

The *Publish Service* is an ICM service that allows you to schedule and run selected Informatica workflows and stored procedures with the ICM service framework. You can use the Publish Service to publish data from ICM to CRM, to import data into ICM, or to export data to other applications. You can run the Publish Service either by itself from within the ICM UI or as part of an ICM service batch.

You can use the Publish Service with custom Informatica workflows that are built to meet your company's requirements. Consequently, you control the details of what data is manipulated. You create and test your own workflows in Informatica, then set up a Publish Service configuration file that references those workflows. The Publish Service does multi-phase processing of workflows, which allows sequential control of workflows as needed.

Process of Setting Up and Running the Publish Service

This topic describes the process of setting up and running the Publish Service. To manage the Publish Service, you perform the following tasks.

1. Set up your Informatica instance to use the Publish Service. See "Setting Up Informatica" on page 130.
2. (Optional) Edit your ICM instance's Informatica properties file. See "Editing the Informatica Properties" on page 130.
3. Set up your Publish Service configuration file. See "Preparing a Publish Service Configuration File" on page 130.
4. Run the Publish Service in one of the following ways:
   - Singly through the ICM UI. See "Running the Publish Service Through the ICM UI" on page 133.
   - As part of a service batch. See "Running the Publish Service in a Service Batch" on page 133.

Setting Up Informatica

To use the Publish Service, you must first set up Informatica. To set up Informatica, follow this procedure.

This task is a step in "Process of Setting Up and Running the Publish Service" on page 129.

To set up Informatica

- Install and configure Informatica. For information, see the topics on Informatica PowerCenter in *Siebel Incentive Compensation Management Installation Guide for UNIX* or *Siebel Incentive Compensation Management Installation Guide for Microsoft Windows*.

Editing the Informatica Properties

ICM's Informatica properties file includes a property specifying the Informatica repository folder that stores the workflows the Publish Service will use. The default setting is siebel_icm_publish. A folder with this name is in the repository included with your ICM distribution. If you store the workflows in a different repository folder, you can change the property's setting to point to that folder.

This task is a step in "Process of Setting Up and Running the Publish Service" on page 129.

To edit the Informatica properties

1. Navigate to the <STAGING>/deploy/informatica directory.

   For <STAGING>, substitute the directory where the ICM files are configured before deployment.

2. In a text editor, open the infa.properties file.
3. Locate the infa.repository.publish.folder.name property, and change its value to the name of the Informatica repository folder in which you are storing the Publish Service workflows.
4. Stop the ICM application server.
5. Navigate to the <STAGING>/deploy directory.
6. Run the following command:

   ```
   ant fast-deploy
   ```

7. Restart the ICM application server.

Preparing a Publish Service Configuration File

A sample publish configuration file is included in the distribution. This file includes configuration settings for the Publish Service and comments describing the XML formatting used by the Publish Service. The default configuration file runs two sample workflows included in the ICM repository. Modify this file as necessary to meet your company's requirements.

This task is a step in "Process of Setting Up and Running the Publish Service" on page 129.

To prepare a Publish Service configuration file

1 Navigate to the config/publishConfig directory.

2 In a text editor, open the publishServiceConfig.xml file.

3 Change the header configuration settings as needed.

Header configuration is defined by the following code line:

```
<icmPublishService name="Publish Service Name" closedPeriodSwitch="openOnly">
```

Header settings are described in the table that follows.

Switch	Comments
name	Can be set to any value. This name is used for reference in the service log.
closedPeriodSwitch	Defines the periods in which the Publish Service can run. Settings are as follows: ■ **openOnly.** Stops the Publish Service when it tries to run in a closed period. ■ **closedOnly.** Stops the Publish Service when it tries to run in an open period. ■ **(no setting, specified as "").** The Publish Service can run in open and closed periods.

4 Specify the Informatica workflows that the Publish Service will run.

Each workflow is identified by the following code lines:

```
<workflow type="wf" name="wf_publish_reading_type" phase="1"
paramFile="e:\wf_publish_reading_type_params.txt">
  <session name="sess_publish_reading_type"/>
</workflow>
```

Workflow settings are described in the table that follows.

Setting	Comments
type	wf represents an Informatica workflow.
name	Name of the Informatica workflow to be run. Must match the name of the workflow as it appears in the Informatica Workflow Manager.
phase	The Publish Service phase in which the workflow is to be run. Valid values are 1, 2, and 3. The phases are functionally equivalent. They allow you to control the order in which the Publish Service runs workflows. For example, if you want to run Workflow X before Workflow Y, you can assign Workflow X to phase 1 and Workflow Y to phase 2.

Setting	Comments
paramFile	Location on the Informatica server of the parameter file to be used when running this workflow.
session name	Name of the Informatica session in which the workflow resides. Must match the name of the session as it appears in the Informatica Workflow Manager. For workflows with multiple sessions, you can use a comma-delimited set of session names.

5 Specify the Informatica stored procedures that the Publish Service will run.

Each stored procedure is identified by the following code line:

```
<workflow type="sp" name="publish_locale_test" dbConnection="transaction" phase="1"
/>
```

Stored procedure settings are described in the table that follows.

Setting	Comments
type	sp represents an Informatica stored procedure.
name	Name of the stored procedure to be run.
dbConnection	The ICM database where the stored procedure is located. Valid values are transaction and analytics.
phase	The Publish Service phase in which the stored procedure is to be run. Valid values are 1, 2, and 3. The phases are functionally equivalent. They allow you to control the order in which the Publish Service runs stored procedures. For example, if you want to run Stored Procedure A before Stored Procedure B, you can assign Stored Procedure A to phase 1 and Stored Procedure B to phase 2.

6 Specify the cache flush entities for the Publish Service.

This feature is for use with workflows that update ICM transaction database tables. Each entity listed will have its ICM database cache cleared after the workflow runs. Each cache flush entity is identified by the following code line:

```
<cacheFlushEntity type="credit" name="com.motiva.ce.credit.CreditImpl" />
```

Cache flush entity settings are described in the table that follows.

Setting	Comments
type	Not used. Included for reference only.
name	Entity name for a table, used by the Publish Service when clearing out its cache. You can determine the entity name for an ICM table from the ICM database mapping definitions in the <DEPLOYMENT>/etc/mapping directory, where <DEPLOYMENT> is the root directory for your ICM installation.

NOTE: If you are not updating any ICM transaction database tables, you can remove the cacheFlushEntity entries from the configuration file.

7 Save and close the publishServiceConfig.xml file.

Running the Publish Service Through the ICM UI

You can run the Publish Service by itself, through ICM's user interface, follow this procedure. To run the Publish Service through ICM's user interface, follow this procedure.

This task is a step in "Process of Setting Up and Running the Publish Service" on page 129.

To run the Publish Service through the ICM UI

- You run the Publish Service through ICM's user interface in the same way you would run any service. For information, see the topic on running a service in *Siebel Incentive Compensation Management Administration Guide*.

  When you run the Publish Service, note the following steps:

  a To navigate to the services list that contains the Publish Service, choose Master Control > Import and Export services.

  b When prompted for a Publish Configuration File, specify the configuration file that you set up in "Preparing a Publish Service Configuration File" on page 130.

Running the Publish Service in a Service Batch

To run the Publish Service in a service batch, you set up a service batch that includes the Publish Service, then run that service batch. To set up and run the Publish Service in a service batch, follow these procedures.

This task is a step in "Process of Setting Up and Running the Publish Service" on page 129.

To add the Publish Service to a service batch

- You include the Publish Service in a service batch in the same way you would include any service. For information, see the following topics:
  - **Through the ICM UI.** See "Creating a Service Batch with the Service Batch Framework" on page 140.
  - **At a command line.** See "Process of Setting Up the Service Launcher" on page 141.

  For an example Publish Service item in a service batch file, see "Service Batch Item for the Publish Service" on page 134.

To run a service batch that includes the Publish Service

- You run a service batch that includes the Publish Service in the same way you would run any service batch. You can do this either with the Service Batch Framework or with the Service Launcher. For information, see the topic on running a service batch in *Siebel Incentive Compensation Management Administration Guide*.

Service Batch Item for the Publish Service

The following code fragment is a sample service batch file item for the Publish Service. For information about setting up a service batch file, see "Creating or Editing a Service Batch File" on page 146.

```
<serviceBatchItem>
<input>e:/staging/Siebel-782/config/publishConfig/publishServiceConfig.xml</input>
   <event name="post_finish"></event>
   <period absolutePeriodNumber="37" periodNumber="1">
      <calendarYear code="FY2006"/>
   </period>
   <serviceDefinition code="PublishService"/>
</serviceBatchItem>
```

The <input> element specifies the Publish Service configuration file that the service batch will use. The directory path in the <input> element refers to an absolute path on the ICM application server.

20 Managing Services and Service Batches

This chapter describes how to create and manage batches of services. It contains the following topics:

- "About Services" on page 135
- "Setting the Services Logging Level" on page 138
- "About Service Batches" on page 139
- "Process of Setting Up the Service Batch Framework" on page 140
- "Process of Setting Up the Service Launcher" on page 141

About Services

Service is a general term for any of ICM's processing functions, including data import, data export, crediting, plan calculation, and period closing. Services in ICM are classified as follows:

- Import Services
- Export Services
- Processing Services

For more information about specific services, and descriptions of the internal processes of some key services, see the chapter on running services and service batches in *Siebel Incentive Compensation Management Administration Guide*.

Import Services

An *import service* is an ICM function that extracts data records from an XML file and copies those records to the relevant ICM database tables. The XML file is generated by an external software system, which could be a third-party application or another instance of ICM.

ICM uses standard import specifications to bring data from external systems into its database. These services can import most entities that are contained in ICM. For a description of how these services process data, see the topic on service processes in *Siebel Incentive Compensation Management Administration Guide*.

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. For a list of valid XML data files and their structures, see Appendix E, "Open Integration Framework Reference."

You can identify the fields that are required for importing by referring to the XSD or DTD files. Required fields have a Use attribute of Required. The XSD and DTD files are located in the <STAGING>/etc/schema/migration directory.

The following topic describes a key import service.

Import Employees Service

This service imports new employee records and updates existing records, based on data in an external import file.

To associate an employee with a supervisor, the supervisor record must exist in the database. Therefore, 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. For example, import executives first, managers second, and so on.

Associating job codes with employees requires that job codes exist in the system. Thus, you should run this service after the Import Job Code service.

Export Services

An *export service* is an ICM function that extracts data records from the ICM database tables and copies those records to an XML file. The XML file can later be imported into an external software system, which could be a third-party application or another instance of ICM.

ICM uses standard export specifications to make data from its database available to external systems. These services can export most entities that are contained in ICM.

Processing Services

A *processing service* is any ICM function that processes data internally within the application. This topic describes some key processing services.

Sales Crediting Service

Any incentive plan that uses transaction records uses this service to generate and distribute credit records. For a description of how this service processes data, see the topic on service processes in *Siebel Incentive Compensation Management Administration Guide*.

The Sales Crediting service can be run in either *incremental mode* or *full batch mode*. Incremental mode processes only those transaction lines and events for which no credits have yet been generated. Full batch mode processes all transaction events that have not yet been closed by the Close Period service. This includes transactions that have already generated credits, even if earning calculations and summarizations have already been calculated in the current period. Full batch mode only excludes transaction events that have been credited and paid in prior processing periods. Those events are marked as “closed” by the Close Period service.

In most cases, end users will use incremental mode, because this speeds up processing. It allows end users to credit only those transactions, transaction adjustments, and so on, that are newly entered. Credits should be processed in full batch mode when significant changes to distribution rules or credit rules are made in the middle of a processing period, and those changes will affect transactions that were processed according to the old rules.

Rollup Service

This service handles measures that roll credits up through an organization hierarchy. If measures exist to roll up goal data, this service also rolls up goal records. For a description of how this service processes data, see the topic on service processes in *Siebel Incentive Compensation Management Administration Guide*.

Cumulate Service

This service handles measures that cumulate credit records. If measures exist to cumulate goals, this service also cumulates goal records.

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

For a description of how this service processes data, see the topic on service processes in *Siebel Incentive Compensation Management Administration Guide*.

Plan Eligibility Service

This service determines which participants are eligible for which plans.

Earning Calculation Service

The Earning Calculation service is part of a series of services that process incentive plans. The Plan Eligibility Service matches participants to plans, the Earning Calculation service calculates earnings for each participant, and the Earning Summarization service condenses the earnings into summarized earnings.

For a description of how this service processes data, see the topic on service processes in *Siebel Incentive Compensation Management Administration Guide*.

Earning Summarization Service

For each payment group, this service generates a summarized earning that is the sum of the earnings for that payment group. You can run this service for one plan or for all plans. Likewise, you can run it for one participant or for all participants.

Trial Payment Calculation Service

This service takes summarized earnings, period beginning balances, and payments made in the period as input. From these, the service calculates the payment amount necessary to arrive at a balance of 0.

Finalize Payment Service

This service takes trial payments as input. It links each trial payment to an actual payment, and creates a separate payment record.

Close Period Service

This service finalizes all processes for the current period. Run this process only if the following statements are true:

- Transactions, credits, and earning 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 distribution rules, formulas, or other plan components.

This process accomplishes the following tasks:

- Updates every participant's balance data to reflect earnings and earning adjustments.
- Updates transaction events and marks events that has been credited and paid as "closed."
- Deletes temporary tables 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 Service

This service copies the transactional database to the analytics database. This allows the analytics functions to access data for the current period without referring directly to the main database. You can run this service at any time during a period, but it is generally useful only at the end of a period, when all transactions have been processed and all earnings assigned.

Setting the Services Logging Level

When an end user launches a service manually, the system offers a choice of detail settings to log processes and errors when the service is running. Each time an end user accesses a service, the system resets the logging level for the services to the default. You can define the default logging level by following this procedure.

To set the services logging level

1. Open a command prompt and navigate to the following directory:

   `<STAGING>/config/`

   For `<STAGING>`, substitute the user-defined temporary location where configuration of ICM occurs before deployment.

2. Open the SiebelICMConfig.xml.in file in a text editor.
3. Locate the service.logging.default property and set the property to one of the logging level settings described in Table 19 on page 205.
4. Save the file and close it.
5. Navigate to the <STAGING>/deploy directory and run the following command:

```
ant deploy-config
```

About Service Batches

A *service batch* is an ordered list of services that can be launched and monitored as a single entity. Administrators use service batches to launch multiple services at one time instead of manually launching the services one at a time. The services run serially; that is, one at a time.

A service batch is contained in a *service batch file*, an XML file that runs a group of services in a specific order. Thus, you can edit a service batch with an XML editor. For more information, see "Creating or Editing a Service Batch File" on page 146.

You can use the following ICM functions to launch a batch of services:

- Service Batch Framework
- Service Launcher

A service batch that contains at least one retroactive service is called a *retroactive service batch*. ICM generates retroactive service batches automatically. You must create non-retroactive service batches manually. For more information about retroactive service batches, see the chapter on retroactive processing in *Siebel Incentive Compensation Management Administration Guide*.

Service Batch Framework

The *Service Batch Framework* is a part of the ICM user interface that serializes and monitors the running of multiple services. The Service Batch Framework is persistent in the ICM database. A service batch can be exported and imported between operating units and between different instances of the application.

Service Launcher

The *Service Launcher* is an alternative to the Service Batch Framework. Where the Service Batch Framework has a user interface, the Service Launcher is a command line utility. The Service Launcher allows administrators to automatically launch services in a specified order and to stop these services based on the number and type of errors encountered. The Service Batch Framework is not persistent in the ICM database.

The Service Launcher is good for running larger import files of sales transactions. If they are too big, they do not run properly through the Service Batch Framework, but they do run properly through the Service Launcher.

You can use this feature to import large amounts of data, such as transactions, every night. Additionally, you can run all the services associated with crediting and earning calculation overnight, so the compensation administrator can review the results and make any necessary changes during normal working hours.

A Service Launcher file can be set up for a single period, for multiple periods, or to run the same service multiple times. Examples of Service Launcher setup files are in the main level of your ICM install directory.

Process of Setting Up the Service Batch Framework

This topic describes the process for creating and managing services batches that users can run from the ICM UI. You perform the tasks in the order in which they are listed, as follows:

1. "Creating a Service Batch with the Service Batch Framework" on page 140.
2. "Importing a Service Batch" on page 141.
3. Running the service batch from within the ICM UI. For information, see the topic on running service batches in *Siebel Incentive Compensation Management Administration Guide*.

Creating a Service Batch with the Service Batch Framework

Use this procedure to create a service batch XML file with the Service Batch Framework.

NOTE: Only processing services are available with this feature. If you want to include export or import services in the service batch, you can edit the XML file later. See "Creating or Editing a Service Batch File" on page 146.

This task is a step in "Process of Setting Up the Service Batch Framework" on page 140.

To create a service batch with the Service Batch Framework

1. Navigate to Master Control > Service Batches.
2. Click the Create New Service Batch link.
3. Complete the fields, as needed. Some fields are described in the following table.

Field	Comments
Code	Batch code for this retroactive service batch.
Launch As User	Launch the services as this user.
Launch As User Password	Launch the services as this user password.
Halt On Service Error	Select to stop the processing of the service batch if a service encounters an error.
Period Range	Range of periods for which the services in the batch are processed.

4 In the Include These Services form, select the check box for each service you want to include in the service batch.

5 Click Save.

Importing a Service Batch

Use this procedure to import a service batch XML file you have created.

CAUTION: The import may fail if the amount of data is more than can be streamed from the Web server to the application server. In these cases, it is recommended that you import with the Service Launcher instead. For more information, see the chapter on operating unit exports and imports in *Siebel Incentive Compensation Management Administration Guide*.

This task is a step in "Process of Setting Up the Service Batch Framework" on page 140.

To import a service batch

1 Navigate to Master Control > Service Batches.

2 Click Browse and navigate to the service batch XML file you want to import.

3 Select the service batch XML file and click Open.

The system redisplays the Service Batches screen with the name and path of the selected XML file displayed in the Service Batch File for Import field.

4 Click Import Service Batch.

ICM imports the XML file and creates a new service batch.

5 Click the service batch's View icon to display the contents of the batch.

NOTE: You can import identical service batches repeatedly. The system allows you to import multiple service batch XML files with the same code by adding a numerical suffix to the code in the form of _n. This makes it possible to do standard nightly processing implemented with OS-level chron or AT jobs.

Process of Setting Up the Service Launcher

This topic describes the process for creating and managing services batches that users can run from a command prompt. You perform the tasks in the order in which they are listed, as follows:

1 "Setting Up a Service Launcher Properties File" on page 142

2 "Creating or Editing a Service Batch File" on page 146

3 Running a service batch from a command line with the Service Launcher. For information, see the topic on running a service batch in *Siebel Incentive Compensation Management Administration Guide*.

Setting Up a Service Launcher Properties File

A *service launcher properties file* launches services in a specified order, and can stop these services depending on the number and type of errors encountered. The service launcher properties file defines the services to run, and the OU and calendar periods in which to run them.

To set up a service launcher properties file, follow this procedure.

This task is a step in "Process of Setting Up the Service Launcher" on page 141.

To set up a service launcher properties file

1. Open a command prompt and navigate to the following directory:

   `//<STAGING>/deploy/servicelauncher`

2. Open one of the following service launcher properties files:

   ```
   servicelauncher.properties
   servicelauncher.crm.interface.properties
   servicelauncher.multiperiod.properties
   servicelauncher.ouimport.properties
   servicelauncher.repeat.properties
   ```

   These are sample service launcher properties files. The files contain instructions on how to proceed. Each of these files contains a different list of services to run.

3. Edit the values, as needed.

   For information about the properties in these files, see "About the Service Launcher Properties" on page 142.

4. Save the file and close it.

NOTE: You can also copy, edit, or create your own service launcher properties files. If you create one, you must run ant fast-deploy to install it.

About the Service Launcher Properties

A service launcher properties file contains the following properties:

- **service.runas.username.** This is the operating unit administrator user name. If this operating unit administrator has access to more than one operating unit, the services will be run on the last operating unit that this user logged in to.
- **service.runas.password.** This is the operating unit administrator password associated with the user name identified in the preceding property.
- **service.runas.scenario.** (Optional) This is the code of the scenario for which you want to run services. The code of the scenario must have been created previously.
- **service.X.yearCode.** Where 'X' stands for the index (execution order) of the service, starting with 1 and incrementing by 1. You must specify the yearCode for the first index but it is optional for other indexes.

- **service.X.periodNumber.** Where 'X' stands for the index (execution order) of the service, starting with 1 and incrementing by 1. You must specify the periodNumber for the first index but it is optional for other indexes.
- **service.X.serviceCode.** One of the following valid values:

ChannelPartnerImport	PeriodCloseService
CostCenterImport	PlanEligibilityService
CreditImport	PlanEntityImport
CumulateService	PlanImport
CustomerImport	ProductImport
EarningCalculationService	PurgePeriodDataService
EarningSummarizationService	ResetScenarioService
EmployeeImport	RollupService
ExchangeRateImport	RuleSetImport
FinalizePaymentService	SalaryGradeImport
FormulaImport	SalesCreditingService
GoalImport	SalesTransactionImport
JobImport	SetupEntityImport
LocationImport	SiebelCRMExtractService
MatrixCalcImport	StepCalcImport
OperatingUnitExport	TerritoryImport
OperatingUnitImport	ThresholdCalcImport
OrganizationImport	TrialPaymentCalculationService
PaymentImport	UpdateAnalyticsService

- **service.X.input.** Represents an optional filename or input text for the service.

  The input requirements for various services are listed in the following table.

Service	Input Requirements
SalesCreditingService	FULL_BATCH or INCREMENTAL. Do not leave blank.
Any Plan/Earning service: ■ PlanEligibilityService ■ EarningCalculationService ■ EarningSummarizationService	At a minimum, these services require the | character. You can also specify a comma-delimited list of plan codes and participant codes, with the two lists separated by the | character, for example: `Plan001,Plan002|Emp001,Emp002` This runs the service for plans Plan001 and Plan002 and participants Emp001 and Emp002 only.
OperatingUnitImport	ID of a migration set.

Service	Input Requirements
Other Import services	Full pathname of the file to be imported. The file must be present on the machine where the Siebel ICM application server is running. Alternatively, EmployeeImport, OrganizationImport, and JobImport can take a migration set ID as input. If you specify a migration set ID, the import service reads that migration set's XML code for the entity to import.

SiebelCRMExtractService	This service has properties in addition to input, as follows:
	
	■ **input.** Name of the configuration file to use—for example, siebelInterfaceConfig.xml. This is the sample file included with the distribution.
	
	■ **extractMode.** FULL or INCREMENTAL. If not specified, defaults to FULL.
	
	■ **beginDate.** MM/dd/yy [hh:mm:ss]. Required if extractMode=INCREMENTAL.
	
	Example of a servicelauncher.properties listing for this service:
	
	`service.X.serviceCode=SiebelCRMExtractService`
	`service.X.input=myCustomSiebelInterfaceConfig.xml`
	`service.X.extractMode=INCREMENTAL`
	`service.X.beginDate=11/01/04`

- **service.X.haltOnLauncherError.** An optional parameter that tells the Service Launcher 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, its haltOnLauncherError remains unchanged from its previous value.
- **service.X.haltOnServiceError.** An optional parameter that tells the Service Launcher 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.

- **service.X.smartPolling.** An optional parameter that allows smart polling. The Service Launcher polls (checks) the Service Manager to determine whether a service is completed.

  If you disable smart polling (the property is set to false), the Service Launcher performs this polling every 30 seconds. If the service runs for more than a few minutes, this frequent checking slows the service's performance. If omitted, the property is set to false.

  If you enable smart polling (the property is set to true), the Service Launcher calculates a polling interval based on the rate of items processed per second. The rate is roughly equivalent to the number of seconds it will take for the service to process 10% of the items. As the service runs, the Service Launcher adjusts the polling interval according to the rate at which the service processes the items.

  It is recommended that you enable smart polling for services expected to run for more than 30 minutes. You can set this property independently for each service in the properties file. If it is not set, it defaults to false.

  You can set the smartPolling property independently or in coordination with the pollInterval property. See "Examples of Smart Polling" on page 145.

- **service.X.pollInterval.** An optional parameter that tells the Service Launcher how often to poll (check) the Service Manager to determine whether a service is completed. Its default value is 30, in units of seconds. If smart polling is enabled, the Service Launcher uses the value of pollInterval as the initial polling interval until the Service Launcher has collected enough data to decide how often to poll.

  The pollInterval property can work independently or in coordination with the smartPolling property. See "Examples of Smart Polling" on page 145.

Examples of Smart Polling

The pollInterval and smartPolling properties avoid the constant checking (currently every 10 seconds) that the Service Launcher performs to determine whether the currently running service is finished. Because the entire application must pause for the check, such frequent checking can have cumulative performance consequences.

You can set pollInteval independently of smartPolling. For example, to poll the service every five minutes (300 seconds) regardless of its rate of process, set pollInteval = 300, and omit smartPolling or set it to false.

Similarly, you can set smartPolling independently of pollInterval. If you set smartPolling to true and omit pollInterval, the Service Launcher uses 30 as the first value of pollInterval but calculates a more appropriate value when the service starts running and a rate of processing can be determined.

You can also use the pollInterval and smartPolling properties together, as in the examples that follow.

```
service.1.serviceCode=SalesCreditingService
service.1.smartPolling=true
service.1.pollInterval=120
```

In the preceding example, smart polling is enabled, with an initial polling interval of 120 seconds. This means that until the Sales Crediting service starts running, the Service Launcher polls the service manager every 2 minutes.

```
service.2.RollupService
service.2.smartPolling=false
service.2.pollInterval=300
```

In the preceding example, smart polling is disabled, and the service launcher polls the service manager once every 300 seconds.

```
service.3.CumulateService
```

In the preceding example, smart polling is disabled by default, and the pollInterval has a value of 30 by default. This means that the Service Launcher polls the Service Manager once every 30 seconds.

The pollInterval and smartPolling properties allow enhanced progress reporting. Every time the Service Launcher polls a service, it prints a snapshot of the service to the console. If smart polling is enabled, the Service Launcher reports every time it adjusts the polling interval, and tries to project the end time and date of the service.

Creating or Editing a Service Batch File

There are several ways to create a service batch file, as follows:

- From a migration set. For information about exporting and importing operating unit data sets, see the chapter on exporting and importing operating units in *Siebel Incentive Compensation Management Administration Guide*.
- From a retroactive revision. See the chapter on retroactive processing in *Siebel Incentive Compensation Management Administration Guide*.
- By building it from services you select in the ICM user interface. See "Creating a Service Batch with the Service Batch Framework" on page 140.
- With a text editor.

To create a service batch file with a text editor, you must be familiar with XML. This approach is *not* recommended. You can use the procedure that follows to create a new service batch, or to edit an existing one.

This task is a step in "Process of Setting Up the Service Launcher" on page 141.

To create or edit a service batch file

1. In a text or XML editor, create a new service batch file, or open an existing one. Note the following information:
   - There is no required path or directory location for service batch files. In most ICM installations, these files are placed in the following location:

     `<STAGING>/test/servicetest/config`

     For `<STAGING>`, substitute the user-defined temporary location where configuration of ICM occurs before deployment.

- Guidelines for formatting a service batch file are listed in "Guidelines for Service Batch XML File Format" on page 147. The file must conform to the format of <DEPLOYMENT>/etc/schema/serviceBatch.xsd. A service batch schema is also reproduced in Appendix F, "XML Service Batch Schema."

2 Save the file and close it.

Guidelines for Service Batch XML File Format

When creating or editing a service batch XML file, observe the following guidelines:

- **Case.** XML element and attribute names are case-sensitive.
- **File contents.** A service batch XML file contains the following elements:
  - Username and password under which to run the batch.
  - Beginning and ending periods of the batch. Services in a batch can span a range of periods and can even skip periods.
  - The list of services to run.
- **Services order.** The services are run in the order in which they are listed in the file.
- **Items.** A service's entry in a service batch is called a *service batch item*. Each service batch item includes the following information:
  - Period for which you want to run the service.
  - Input for the service—for example, the XML file to use for an import service.
  - Whether to halt the batch if the service encounters an error.
  - (Optional) Whether to halt the batch if the service reports "No Items to Process."
- **Service inputs.** Different services require different inputs. For descriptions of inputs for some services, see "Processing Services" on page 136 and the topic on service processes in *Siebel Incentive Compensation Management Administration Guide*.

21 Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Integration

This chapter describes the integration of Oracle's Siebel Customer Relationship Management (CRM) product family to Siebel ICM. It contains the following topics:

- "About Siebel CRM-to-Siebel ICM Integration" on page 149
- "Process of Running Siebel CRM-to-Siebel ICM Integration" on page 151
- "Reference Implementation Field Mappings" on page 159
- "About Data Volume and Performance" on page 161
- "Troubleshooting Siebel CRM-to-Siebel ICM Integration" on page 162

About Siebel CRM-to-Siebel ICM Integration

Implementations in which Siebel CRM applications and Siebel Incentive Compensation Management 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 for common reference data, including, but not limited to, Employees, Products, and Jobs. This data will be extracted from Siebel CRM to synchronize with Siebel ICM. Changes in Siebel ICM data will not be copied or loaded back into Siebel CRM.

A set of reference implementations of this 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.

Extract, Transform, or Load

The Siebel ICM Interface should be considered a reference implementation, and may need some modification in the field. Integration follows the Extract/Transform/Load pattern. The integration extracts data from Siebel CRM in XML format, then transforms the XML to a form suitable for import into Siebel ICM. The extract and transform steps are incorporated into a single Java utility, built around the Siebel Java Data Bean API. The Load step is separate. Here is an outline of the process:

- **Extract.** The extract step is implemented as a Siebel workflow consisting of the EAI Siebel Adaptor business service's Query method and the EAI XML Converter business service. This workflow is shipped with Siebel ICM. The workflow requires two input parameters—a search specification and the name of the Integration Object to query. The workflow's output is a string containing an XML document that represents the output of the query method. This output is passed to the transform step.

- **Transform.** The transform step applies one or more style sheets to the XML, resulting in one or more output XML documents. These documents can be saved as part of an export set suitable for import using the Operating Unit Import Service, or as individual documents suitable for import through individual import services.
- **Load.** The load step is covered by Siebel ICM's Operating Unit Import service and other Import Services.

Search Specification

The search specification string is fully configurable, as is the integration object name and the workflow name itself, allowing you to override workflows, integration objects, and search specifications.

Full Extract Compared With Incremental Extract

You can use the interface framework to perform full extracts, where every record in the business component is extracted according to the search specification. Alternatively, you may configure the interface framework to perform an incremental extract, where the search specification is modified to include a clause specifying that only those records updated since a given date be returned by the extract.

Data Loads into Siebel ICM

You can configure the interface to create export sets, individual XML files, or a combination of them. By default, the export set is created for operating unit Siebel, period 1. Export sets are loaded into Siebel ICM through the Operating Unit Import service.

Individual XML files are loaded into Siebel ICM through the appropriate Import Service for the entity represented in XML. For example, if the individual XML file has been created from the Employee Integration Object, it should be loaded into Siebel ICM using the Employee Import Service.

For large volumes of data (5,000 records or more), it is recommended that you use the individual XML file approach, because the entity-based Import Services are tuned for large-volume data imports.

Reference Implementation Entities

The reference implementation includes the following entities:

- Employee
- Position
- Account
- Product

For mappings of fields to XML elements for each of these entities, see "Reference Implementation Field Mappings" on page 159.

You can incorporate additional entities into the interface by adding entries to the interface configuration file. For more information, see "Customizing Siebel ICM to Receive Siebel CRM Data" on page 154.

Process of Running Siebel CRM-to-Siebel ICM Integration

This topic describes the process of running Siebel CRM-to-Siebel ICM integration.

1. "Installing the Integration Workflow into Siebel CRM" on page 151
2. "Installing the Integration Objects into Siebel CRM" on page 152
3. "Setting Siebel ICM Configuration Properties" on page 153
4. "Customizing Siebel ICM to Receive Siebel CRM Data" on page 154
5. Extracting the data from Siebel CRM in one of the following ways:
   - "Extracting Siebel CRM Data with Siebel ICM" on page 157
   - "Extracting Siebel CRM Data with a Service Batch File" on page 159
6. "Loading the Siebel CRM Extract into Siebel ICM" on page 159

Installing the Integration Workflow into Siebel CRM

The system extracts Siebel ICM-bound data from Siebel CRM by running the workflow titled Incentive Comp Extract Workflow within Siebel CRM. To allow the system to extract this data from Siebel CRM, you must install the workflow into Siebel CRM.

NOTE: Although it is bundled with the ICM product, this workflow is a Siebel CRM component and runs *only* in Siebel CRM, *not* in ICM. The ICM application does not have a workflow engine.

This task is a step in "Process of Running Siebel CRM-to-Siebel ICM Integration" on page 151.

To install the Integration Workflow into Siebel CRM

1. Import the Incentive Comp Extract Workflow.xml file located under the Siebel ICM root directory in the Siebel CRM application subdirectory etc/SiebelTools/Workflows.

   This imports the workflow process named Incentive Comp Extract Workflow.

   - For Siebel CRM 7.5.3, you import the workflow through the Siebel client application connecting to your Siebel Server data source (for example, Siebel FINS or Siebel Call Center).

- For Siebel CRM 7.7 and 7.8, you import the workflow through Oracle's Siebel Tools product, connecting to your Siebel Server data source.

For instructions on importing workflows into Oracle's Siebel applications, see the topic on importing or exporting process definitions in *Siebel Business Process Designer Administration Guide* for your version of Siebel CRM.

2 Activate the workflow process.

For instructions on activating workflows, see the topic on activating workflow processes in *Siebel Business Process Designer Administration Guide* for your version of Siebel CRM.

Installing the Integration Objects into Siebel CRM

To allow the system to extract Siebel ICM-bound data from Siebel CRM, you must install the following integration objects into Siebel CRM.

- Siebel ICM Employee
- Siebel ICM Product
- Siebel ICM Account
- Siebel ICM Position

This task is a step in "Process of Running Siebel CRM-to-Siebel ICM Integration" on page 151.

To install the integration objects into Siebel CRM

1 Start Siebel Tools and connect to your Siebel Server data source.

2 To create ICM integration objects, do the following:

a Navigate to the <DEPLOYMENT>/etc/SiebelTools directory.

NOTE: <DEPLOYMENT> refers to the root directory of your ICM installation.

b Import the following files:

- siebel_icm_employee.sif
- siebel_icm_product.sif
- siebel_icm_account.sif
- siebel_icm_position.sif

For instructions on importing SIF files into Siebel Applications, see the topic on importing object definitions in *Developing and Deploying Siebel Business Applications*.

3 Compile the imported integration objects into the Siebel repository file on the server side.

Setting Siebel ICM Configuration Properties

Certain properties must be set for the Siebel ICM Interface to connect to a Siebel Server and extract data.

This task is a step in "Process of Running Siebel CRM-to-Siebel ICM Integration" on page 151.

To set Siebel ICM configuration properties

1. Log on to the Siebel ICM application server computer.
2. Navigate to the <DEPLOYMENT>/siebel_crm_interface directory.
3. Edit the siebel.crm.properties file, setting the properties for the Siebel CRM connection string, Siebel CRM user name and password, Siebel language, and other connection options, as follows:

```
siebel.connection.string=siebel.TCPIP.none.NONE://<hostname>:<port>/Siebel/
<app>ObjMgr_enu
siebel.user.name=SADMIN
siebel.user.password=SADMIN
siebel.user.language=ENU
siebel.user.encrypted=false

# Default is 600 seconds. Indicates the transaction timeout in seconds on the client
side.
siebel.conmgr.sesstimeout=600

# Default is 2700 seconds. Indicates the transaction timeout in seconds on the server
side.
siebel.conmgr.txtimeout=2700

# Default is 2 with maximum of 500. Indicates the connection pool size.
# Connection pool maintains a set of connections to a specific server process.
siebel.conmgr.poolsize=5

# Indicates the usage of Java Cryptography Extension.
# You can set this to 1 for jce usage and 0 for nonusage.
siebel.conmgr.jce=0

# Number of times to retry
siebel.conmgr.retry=5
```

   The siebel.connection.string property connects to an instance of the Siebel Server.

   - <hostname> is the hostname of the machine running the instance, usually myserver or siebelServer.
   - <port> is the port number.
     - Integration with Siebel CRM 7.5.3 uses port number 2320 (the default gateway server port), and appends the name of the Siebel Server to the end of the string.
     - Integration with Siebel CRM 7.7 uses port number 2321 (the default request broker port) and omits the name of the Siebel Server.

- Integration with Siebel CRM 7.8 uses port number 2321 (the default request broker port) and omits the name of the Siebel Server. This port number is dependent on the Siebel CRM installation and configuration. For complete connect string syntax, see *Siebel Object Interfaces Reference*.

- <app>ObjMgr_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.

An example of a connect string follows:

```
siebel.connection.string= siebel.TCPIP.none.NONE://myserver:2321/siebel/
FINSObjMgr_enu
```

4 Save and close the siebel.crm.properties file.

5 Redeploy Siebel ICM by navigating to the <STAGING>/deploy directory and running the following target:

```
ant fast-deploy
```

6 Restart the application server.

After editing the siebel.crm.properties file and redeploying ICM, you can run the interface.

Customizing Siebel ICM to Receive Siebel CRM Data

You can set up the Siebel ICM interface to extract certain entities and apply desired transformations. No configuration is required, unless you want to extract different integration objects, use a custom workflow, or apply your own transformations to the extracted data.

This task is a step in "Process of Running Siebel CRM-to-Siebel ICM Integration" on page 151.

To customize Siebel ICM to receive Siebel CRM Data

1 Navigate to the /config/siebelInterfaceConfig directory.

2 Locate the siebelInterfaceConfig.xml.in file and open it in an editor.

CAUTION: Make sure you edit the file with the .in extension. If you edit the file with the .xml extension, your changes will be erased the next time you do a build.

You can edit the siebelInterfaceConfig.xml.in file as is, or you can copy it, save it as a different file, and then edit it. When you run the service, you pick the file to use. All Siebel CRM Extract Service configuration files are in the /config/siebelInterfaceConfig directory.

There is also a W3C Schema file included with the distribution that documents the format of this XML file. The schema file is /etc/schema/Siebel/siebelInterfaceConfig.xsd.

NOTE: Schemas for the Employee, Product, Position, and Account (in other words, Customer) Integration Objects are also in the /etc/schema/Siebel/ directory.

3 Modify the attributes in the following portion of the file according to your requirements.

```
<?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.motiva.ce.employee.EmployeeVersionImpl"/>
   </workflowConfig>
   <workflowConfig>   … additional invocations of the workflow …
```

This portion of the siebelInterfaceConfig.xml.in file contains the following elements:

- siebelInterfaceConfig
- workflowConfig

siebelInterfaceConfig

siebelInterfaceConfig is the root element. Table 11 lists this element's attributes.

Table 11. siebelInterfaceConfig Attributes

Attribute	Comments
name	Name of the interface configuration. You can give it any value you like.
connectString	Do not change the value. Will be set by the build.
langCode	Do not change the value. Will be set by the build.
siebelUsername	Do not change the value. Will be set by the build.
siebelPassword	Do not change the value. Will be set by the build.

workflowConfig

workflowConfig invokes the workflow on the Siebel Applications side one time. Table 12 lists this element's attributes.

Table 12. workflowConfig Attributes

Attribute	Required / Optional	Comments
name	Required	Descriptive name with no other significance.
siebelWorkflowName	Required	Name of the Siebel workflow that will be invoked. Default value is Incentive Comp Extract Workflow. Do not change the default unless you have built and activated a custom workflow. See "Installing the Integration Workflow into Siebel CRM" on page 151.
integrationObjectName	Required	Name of the integration object that the workflow will extract. The name should correspond to an integration object present in the Siebel Server repository. **CAUTION:** If you have created a custom integration object, make sure it has been compiled into the repository.
dumpToFile	Optional	Filename to which you can dump the raw data. This is useful if debugging the search specification or preparing to write a new style sheet.

The workflowConfig element contains the following child elements:

- searchSpec
- transform

Multiple workflowConfig child elements and multiple transform child elements can exist within one workflowConfig element.

searchSpec

searchSpec contains a search specification string. This string is passed to the query that extracts data for integrationObjectName. For information about the syntax of search specification strings, see the interfaces reference chapter of *Siebel Object Interfaces Reference*. Only one searchSpec child element may be present within the workflowConfig element.

transform

Set this element to the name of an XSL stylesheet to be applied to the XML representation of the extracted Integration Objects. transform has multiple elements, each of which represents a single transformation to be run against the data extracted by the workflow. One or more transform child elements may be present within the workflowConfig element. Table 13 lists this element's attributes.

Table 13. transform Attributes

Attribute	Comments
addToExportSet	Boolean flag. Set to True to add the transform output to an export set. If True, the entityClassName attribute must also be set.
entityClassName	Entity class name for which you create an export set entry. This attribute should have the value of the entity class you are creating—for example, com.motiva.ce.employee.EmployeeVersionImpl. The operatingUnitImportList.properties file contains the list of valid entityClassNames. This attribute *must* be set if addToExportSet=True.
outputFilename	Filename to which you write output XML. Use this attribute if you are *not* generating XML as part of an export set. Make sure the filename is fully qualified appropriately for your platform; otherwise, it will be written to the current working directory of the application.
stylesheet	Name of the style sheet—for example, employee.xsl. Place stylesheets in the <deployroot>/etc/migration/xsl/siebel directory. Then, deploy them to the <appserver> directory by running the build target deploy-mapping. You do not need to include any path information unless you have created subdirectories under <deployroot>/etc/migration/xsl/siebel. In addition to the reference implementation style sheets, an identity transformation style sheet, called identity.xsl, has been included. You can use this style sheet to see the resulting output of the extract. Its result is different from the dumpToFile attribute listed under "workflowConfig" on page 156. dumpToFile writes the raw data from Siebel Applications, where the identity transformation writes the data after it has been run through the transformation engine (xalan).

Extracting Siebel CRM Data with Siebel ICM

You extract Siebel ICM-bound data from Siebel CRM by running the Siebel CRM Extract Service. This service pulls data from Siebel CRM as XML files.

This task is a step in "Process of Running Siebel CRM-to-Siebel ICM Integration" on page 151.

To extract Siebel CRM data with Siebel ICM

1. In the your Siebel CRM application's Services window, verify that the Siebel Gateway Name Server, Siebel ICM, and Siebel Server services are started.
2. Launch Siebel ICM and log on as a system administrator.
3. On the Siebel ICM home page, click the Services button.
4. In the Processing Services list, click the Siebel CRM Extract link.
5. On the Launch Service screen, in the Log Level field, select the desired logging level.
6. Click Launch Service.
7. On the Siebel CRM Extract Service screen, complete the fields. Some fields are described in the following table.

Field	Comments
CRM Interface Configuration	Select the configuration file set up in "Customizing Siebel ICM to Receive Siebel CRM Data" on page 154. The service extracts data from Siebel CRM according to the specifications in this file.
Extract Mode	Specify a full or incremental extract. If you select Incremental, ICM displays the Extract All Changes Since Date field.
Extract All Changes Since Date	Beginning date for an incremental extract.
Connect String	If you want to log on to Siebel CRM with a different connect string from the one in the siebel.crm.properties file, enter it here.
Siebel Username	If you want to log on to Siebel CRM with a different user name from the one used to log on to Siebel ICM, enter it here.
Siebel Password	If you want to log on to Siebel CRM with a different password from the one used to log on to Siebel ICM, enter it here.

8. Click OK to start the service.

   The Siebel CRM Extract Service does the following:

   - Logs on to Siebel CRM and extracts records, optionally applying a search spec to filter the records retrieved by the extract.
   - Applies one or more XSL transformations to the extracted data, generating one or more XML files. The service may generate a migration set, or it may generate one or more stand-alone XML files.

Extracting Siebel CRM Data with a Service Batch File

You can run the Siebel CRM Extract Service, which extracts data from Siebel CRM, as part of a service batch XML file that you create manually. For details on how to construct and run a service batch file, see Chapter 20, "Managing Services and Service Batches."

Loading the Siebel CRM Extract into Siebel ICM

The Siebel CRM extract XML files generated by the Siebel CRM Extract Service can now be loaded into Siebel ICM through the appropriate import services. For instructions on running ICM services, see the chapter about running services and service batches in *Siebel Incentive Compensation Management Administration Guide*.

Reference Implementation Field Mappings

This topic contains mappings of fields to XML elements for each of the entities in the reference implementation.

Table 14 displays the reference implementation field mappings for the Employee entity. XML element names are in migration XML format.

Table 14. Employee Entity Field Mappings

Employee Integration Object	Employee XML Element Name
LoginName	code
Id	externalRefID
FirstName	firstName
LastName	lastName
MiddleName	middleName
EMailAddr	emailAddress
Fax	officeFax
Phone	officePhone
HomePhone	homePhone
HireDate	hireDate
TerminationDate	terminationDate
EmployeeEndDate	employeeJobVersion.endDate
EmployeeStartDate	employeeJobVersion.startDate

Table 14. Employee Entity Field Mappings

Employee Integration Object	Employee XML Element Name
OrganizationId	employeeJobVersion.organization.code
PositionId	employeeJobVersion.job.code

Table 15 displays the reference implementation field mappings for the Account entity. XML element names are in migration XML format.

Table 15. Account Entity Field Mappings

Account Lite Integration Object	Customer XML Element Name
Name	code
AccountId	externalRefID
MainPhoneNumber	officePhone
MainFaxNumber	officeFax
BusinessAddress.StreetAddress	address1
BusinessAddress.StreetAddress2	address2
BusinessAddress.City	city
BusinessAddress.County	county
BusinessAddress.State	state
BusinessAddress.PostalCode	zipCode
BusinessAddress.Country	country
AccountStatus	status
CurrencyCode	currency.code
Type	type

Table 16 displays the reference implementation field mappings for the Product entity. XML element names are in migration XML format.

Table 16. Product Entity Field Mappings

Product Integration Object	Product XML Element Name
ProductName	code
Id	externalRefID
Description	description
“Product” (literal)	hierarchyLevelID

Table 17 displays the reference implementation field mappings for the Position entity. XML element names are in migration XML format.

Table 17. Position Entity Field Mappings

Position Integration Object	Job XML Element Name
Name	code
Description	description
Id	externalRefID

About Data Volume and Performance

Some configuration issues involve system performance in relation to data volume. This topic contains additional configuration information for those issues.

Extract 10,000 Records or More from Siebel CRM

If you plan to extract 10,000 or more records from Siebel CRM, you may need to alter the DSMaxFetchArraySize configuration parameter in the Object Manager. If during the extract you see this error message:

```
There were more rows than could be returned. Please refine your query to bring back fewer rows.
```

Then consider changing this parameter to -1, or changing your query (search specification) to bring back fewer rows.

To change the DSMaxFetchArraySize parameter value

1. Navigate to the Site Map.
2. Choose Administration - Server Configuration > Enterprises > Profile Configuration.
3. In the Profiles list, select ServerDataSrc.
4. In the Profile Parameters list, select DSMaxFetchArraySize.
5. Change the Value of DSMaxFetchArraySize to -1.

CAUTION: When you change the DSMaxFetchArraySize configuration parameter, you override a safeguard. Changing this parameter may cause every query to return more than 10,000 records. This can tie up memory and cause performance problems. To avoid this situation, you can change the parameter, run the volume extract, and then change the parameter back to its original or default value.

Load More Than 5,000 Records into Siebel ICM

If you plan to load more than about 5,000 records into Siebel ICM, you may want to generate "legacy" XML as opposed to migration-style XML. This is because the Operating Unit Import service, which loads export sets, is not designed or tuned for loading large numbers of records (more than 5000). It is recommended that you write the Transform style sheet to generate legacy XML and import it through the individual entity import services, such as Employee Import or Customer Import.

For information on generating XML and importing it through the Open Integration Framework, see Appendix E, "Open Integration Framework Reference."

Troubleshooting Siebel CRM-to-Siebel ICM Integration

This topic provides guidelines for resolving Siebel CRM-to-Siebel ICM integration problems.

To address a failure of Siebel ICM to extract data from Siebel CRM, try the solutions in Table 18.

Table 18. Resolving Siebel CRM-to-Siebel ICM Integration Problems

Possible Cause	Diagnostic Steps/Solution
The connect string may be incorrect.	Follow the steps of "Testing the Connect String" on page 163. If the connect string does not work, fix it in the siebel.crm.properties file.
One or more integration objects may be missing from the Siebel CRM SRF file.	Make sure the Integration Object exists in the Siebel CRM SRF file for your CRM instance. If it does not, reinstall the integration objects and recompile the SRF file with the integration objects included.
The integration workflow may be missing from or inactive in Siebel CRM.	Make sure the integration workflow exists and is Active in your Siebel CRM instance: ■ If the integration workflow does not exist in Siebel CRM, reinstall it. ■ If the integration workflow is not Active, change its status to Active.
Search Spec might not be filtering the records.	Check the syntax of the search spec by using the Workflow Simulator tool in the Siebel CRM client application. See *Siebel Connector for Oracle Applications* or *Siebel Connector for SAP R/3*.
The query may have returned too many results.	It is recommended that you break up large extracts (in excess of 5,000 records) into smaller chunks, or that you remove *any unnecessary* Business Components and Fields from the integration object. See "About Data Volume and Performance" on page 161.

Testing the Connect String

You can test whether the connect string works from within the Siebel ICM user interface.

To test the connect string

1. Launch Siebel ICM and log on as a system administrator.
2. On the Siebel ICM home page, click the Services button.
3. In the Processing Services list, click the Siebel CRM Extract link.
4. On the Launch Service screen, click Launch Service.
5. On the Siebel CRM Extract Service screen, click Test Connection.

   The system attempts to open a connection to the Siebel CRM instance specified in the connect string with the username and password you have entered. The system displays a message below the Connect String field, indicating successful connection or an error.

22 Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Single Sign-On

This chapter describes how to configure single sign-on from Siebel CRM to Siebel ICM. It contains the following topics:

- "About Siebel CRM to Siebel ICM Single Sign-On" on page 165
- "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165

About Siebel CRM to Siebel ICM Single Sign-On

You can configure Siebel CRM and Siebel ICM so that a user can access Siebel ICM directly from within the Siebel CRM application without logging on to Siebel ICM separately. This functionality is called *single sign-on*. It involves mapping Siebel CRM responsibilities to Siebel ICM roles, then creating a screen tab in Siebel CRM to allow access to Siebel ICM. This tab, when selected, authenticates an end user against Siebel ICM with that end user's Siebel CRM user name and password. Upon successful authentication, it launches Siebel ICM in a new browser window.

Process of Configuring Siebel CRM to Siebel ICM Single Sign-On

This topic describes the process for configuring Siebel CRM to Siebel ICM Single Sign-On.

1. "Identifying Siebel CRM Responsibilities for Siebel ICM Access" on page 166
2. "Setting Siebel ICM Properties for Siebel CRM Authentication" on page 166
3. "Mapping Siebel ICM Roles to Siebel CRM Responsibilities" on page 167
4. "Copying Siebel ICM Web Templates to Siebel CRM" on page 168
5. "Creating Siebel ICM Screen Tab Objects" on page 168
6. "Adding an ICM Screen Tab to a Siebel CRM Application" on page 173
7. "Adding Siebel ICM Screen Access to Responsibilities" on page 173
8. "Configuring Siebel CRM to Edit Seed Data" on page 173
9. "Testing the Siebel ICM Tab in Siebel CRM" on page 175

Identifying Siebel CRM Responsibilities for Siebel ICM Access

You must decide which types of Siebel CRM users can see and use the Siebel ICM screen. This involves mapping Siebel CRM responsibilities to Siebel ICM roles.

In Siebel ICM, a *role* is a collection of privileges that specify which areas of the application a user can access. In Siebel CRM, a *responsibility* controls the views to which a user has access. An ICM privilege is roughly analogous to a CRM view. A CRM responsibility maps one-to-one with an ICM role.

Decide which responsibilities in Siebel CRM should be allowed access to Siebel ICM, and what access privileges within ICM each responsibility will have. This involves listing CRM responsibilities that need ICM privileges.

For more information on Siebel ICM roles, see Chapter 5, "Setting Up Security." For more information on Siebel CRM responsibilities, see the chapter on access control in *Security Guide for Siebel Business Applications*.

Setting Siebel ICM Properties for Siebel CRM Authentication

To allow single sign-on, you must set properties so that Siebel ICM can connect to Siebel CRM and use Siebel CRM authentication.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To set Siebel ICM properties for Siebel CRM authentication

1. Navigate to the <STAGING>/siebel_crm_interface directory for your Siebel ICM installation.
2. Open the siebel.crm.properties file in a text editor and set the connect string values. For details, see "Customizing Siebel ICM to Receive Siebel CRM Data" on page 154.

   This allows Siebel CRM and Siebel ICM to connect with each other and share data.
3. Open the edit login-config.xml.in file and uncomment the section for Siebel Authentication.

   This configures Siebel ICM for Siebel CRM authentication.

   For more information about configuring multiple authentication modes, see "About Configuring Multiple Authentication Modes" on page 167.
4. Redeploy and restart ICM.

   For detailed instructions on these steps, see the chapter on Siebel ICM installation in *Siebel Incentive Compensation Management Installation Guide for UNIX* or *Siebel Incentive Compensation Management Installation Guide for Microsoft Windows*.

About Configuring Multiple Authentication Modes

To configure multiple modes it is recommended that you use JAAS. JAAS is the standard set of APIs to use, and it includes the ability to stack authentication modules. Each login module has a snippet of XML that defines it. Each module has a flag (sufficient, required, optional, and so on) that allows the user to authenticate against multiple sources. Depending on the requirements of your company and your ICM implementation, you put the login modules in order, with appropriate flags.

Typically, for ICM you would stack the login modules as follows:

- castor login module with participants=false/admin=true
- YOUR MODULE (LDAP, AD, or other)
- Admin login module for bootstrapping

Mapping Siebel ICM Roles to Siebel CRM Responsibilities

To allow single sign-on, Siebel ICM must contain roles that match the responsibilities that will have access from within Siebel CRM. This will allow Siebel CRM users to access the parts of Siebel ICM they need to do their jobs.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To map Siebel ICM roles and users to Siebel CRM

1. Log on to Siebel ICM as the Operating Unit Administrator.
2. Identify each role corresponding to a CRM responsibility and add the necessary privileges.

   For information about adding roles and privileges, see "Adding Roles and Assigning Privileges" on page 39.

   CAUTION: Each role name must exactly match its corresponding Siebel responsibility name, character for character. If it does not, access to Siebel ICM from Siebel CRM will fail for all users with the non-matching role and responsibility.
3. Launch the Siebel CRM instance referenced by the connect string you set in "Setting Siebel ICM Properties for Siebel CRM Authentication" on page 166.
4. Extract the employee and user and role mapping information from the Siebel CRM instance, using the Siebel CRM Extract Service.
5. Load employees into Siebel ICM.
6. Load users and role mapping information into Siebel ICM.

   For information about importing information from Siebel CRM, see Chapter 21, "Configuring Siebel Customer Relationship Management-to-Siebel Incentive Compensation Management Integration."

   After completing this task, you can log on to Siebel ICM with one of the user accounts imported from Siebel CRM, if you know the password.

Copying Siebel ICM Web Templates to Siebel CRM

Copying Siebel ICM Web templates to Siebel CRM creates parts of the Siebel CRM Web application that are critical to displaying the tab that will access Siebel ICM.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To copy Siebel ICM Web Templates to Siebel CRM

1. Navigate to the <STAGING>/etc/SiebelTools/WEBTEMPL directory.
2. Copy the ICMApplet.swt and ICMView.swt files into your Siebel Server WEBTEMPL directory.

Creating Siebel ICM Screen Tab Objects

This procedure describes how to create the objects that are necessary for a Siebel ICM access screen tab in a Siebel CRM application.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To create Siebel ICM screen tab objects

1. Launch Oracle's Siebel Tools product.
2. Navigate to Business Component and create a new business component record with values as shown in the following table.

Property	Value
Name	ICM SSO
Class	CSSBCVExternalURL
Dirty Reads	TRUE
Log Changes	TRUE
No Delete	TRUE
No Insert	TRUE
No Update	TRUE
No Merge	TRUE

3. Add a field to the newly created business component with values as shown in the following table.

Property	Value
Name	SiebelICM
Calculated Field	TRUE

Property	Value
Value	“SiebelICM”
Type	DTYPE_TEXT
Use Default Sensitivity	TRUE

4 Navigate to Business Objects and create a new business object with values as shown in the following table.

Property	Value
Name	ICM SSO BO
Query List Business Component	Query List

5 Add the following business component to the newly created business object with values as shown in the following table.

Property	Value
Name	ICM SSO

6 Modify the following field for the newly created business object with values as shown in the following table.

Field	Value
Primary Business Component	ICM SSO

7 Navigate to Web Template and create a new Web template with values as shown in the following table.

Property	Value
Name	ICM Applet Template
Type	Applet Template - Specialized

8 Add a Web template file for the newly created Web template with values as shown in the following table.

Property	Value
Name	ICM Applet Template
File Name	ICMApplet.swt

9 Navigate to Applets. Copy the applet named SSO Siebel Answers Applet and set the copied records properties as shown in the following table.

Property	Value
Title	Incentive Compensation Management **NOTE:** You must create a new string or use a title override.
Class	CSSFrameListWeb
Name	SSO Siebel ICM Applet
Business Component	ICM SSO

10 Delete the existing list column named Siebel Answers from the copied applet.

11 Add a new list column with values as shown in the following table.

Property	Value
Name	SiebelICM
Field	SiebelICM
Available	TRUE
Field Retrieval Type	Symbolic URL
HTML Display Mode	Encode Data
HTML List Edit	TRUE
HTML Row Sensitivity	TRUE
HTML Type	Text
Text Alignment	Left
Show in List	TRUE

12 Delete the existing applet Web template named Siebel Answers from the copied applet and create a new one with values as shown in the following table.

Property	Value
Name	ICM Applet Template
Web Template	ICM Applet Template
Sequence	1
Type	Base

13 Add an applet Web template item to the applet Web template with values as shown in the following table.

Property	Value
Name	SiebelICM
Control	SiebelICM
Item Identifier	501
Type	List Item

14 Navigate to Web Template and create a new Web template with values as shown in the following table.

Property	Value
Name	ICM View Web Template
Type	View Template

15 Add a Web template file for the newly created Web template as shown in the following table.

Property	Value
Name	ICM View Web Template
File Name	ICMView.swt

16 Navigate to View and create a new view with values as shown in the following table.

Property	Value
Name	SSO Siebel ICM View
Business Object	SSO Siebel ICM View
Add To History	TRUE

17 Add a view Web template to the newly created view with values as shown in the following table.

Property	Value
Name	SSO Siebel ICM Web Template
Web Template	ICM View Web Template

18 Add a view Web template item to the newly created view Web template with values as shown in the following table.

Property	Value
Name	SSO Siebel ICM Web Template Item
Applet	SSO Siebel ICM Applet
Item Identifier	1
Applet Mode	Base

19 Modify the newly created view record as shown in the following table.

Property	Value
Visibility Applet	SSO Siebel ICM Applet

20 Navigate to Screen and create a new screen with values as shown in the following table.

Property	Value
Name	Siebel ICM Screen
Viewbar Text	ICM **NOTE:** You must create a new string or use a title override.

21 (Optional) Enter a status message on the screen.

22 Add a screen view to the newly created screen with values as shown in the following table.

Property	Value
Name	SSO Siebel ICM View
View	SSO Siebel ICM View
Sequence	1,001,000,000
Display In Page	TRUE
Viewbar Text	ICM **NOTE:** You must create a new string or use a title override.
Display in Site Map	TRUE
Menu Text	ICM **NOTE:** You must create a new string or use a title override.
Type	Aggregate View
Default View	SSO Siebel ICM View

23 Modify the newly created screen as shown in the following table.

Property	Value
Default View	SSO Siebel ICM View

Adding an ICM Screen Tab to a Siebel CRM Application

Determine the list of Oracle's Siebel applications for which to add ICM as a page tab or site map entry. For each application, use the following procedure.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To add an ICM screen tab to a Siebel CRM application

1 Lock the project if necessary.

2 In Page Tab, add Siebel ICM Screen to the Page Tab collection.

3 Set Sequence as an integer representing how far to the right the tab should appear.

It is recommended that you enter a number higher than the number of tabs that are already present.

4 In Screen Menu, add Siebel ICM Screen to the Screen Menu collection to appear on the Site Map.

5 Compile all changes to the appropriate SRF file and restart your Siebel Server.

Adding Siebel ICM Screen Access to Responsibilities

When ICM screen tabs are added to Siebel CRM, you can associate these screen tabs with the responsibilities you want to have access to Siebel ICM. For details, see the chapter on configuring access control in *Security Guide for Siebel Business Applications*.

Configuring Siebel CRM to Edit Seed Data

You configure the Siebel CRM client to edit seed data because editing responsibilities (or seed data) require this parameter be active.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To configure Siebel CRM to edit seed data

1 Add the `/editseeddata` switch to your Siebel CRM application shortcut's properties.

2 Launch the Siebel CRM client application.

3 Navigate to Site Map > Administration - Application > Views.

4 Create a new record for the SSO Siebel ICM View created in Step 16 of "Creating Siebel ICM Screen Tab Objects" on page 168.

NOTE: The view does not appear in the pick list. You must type the name into the field.

5 Add the view to those responsibilities you want to grant access to ICM.

6 Navigate to Site Map > Administration - Integration.

7 From the tab-level drop-down list, select WI Symbolic URL.

8 In the Web Application view, create an entry for SiebelICM.

9 In the Host Administration view, create a new record with field values set as in the following table.

Field	Value
virtual name	ICMHOST
name	[ICMHostServer] or similar "placeholder" value
OR: name	Actual server name of host:port where ICM is running—for example, myICMserver:8080. Typically, ICM is configured to listen on port 8080.

10 In the Symbolic URL view, make sure the URL contains the virtual name "ICMHOST": http://ICMHOST/Siebel/crm_sso.jsp.

11 In the Symbolic URL view, create a new record with field values set as in the following table.

Field	Value
name	business component field name from Tools: "SiebelICM"
web application name	"SiebelICM"
Fixup Name	"Default" means do nothing to the content served from the ICM Web server.
SSO Disposition	IFrame
Web Application Name	"SiebelICM"

12 Add a doPost parameter to the record, with field values set as in the following table.

Field	Value
ArgumentType	Command
ArgumentValue	"PostRequest"
Append as Argument	True (checked)

13 Add a j_username parameter to the record, with field values set as in the following table.

Field	Value
ArgumentType	Command
ArgumentValue	UseSiebelLoginId
Append as Argument	True (checked)

14 Add a j_password parameter to the record, with field values set as in the following table.

Field	Value
ArgumentType	Command
ArgumentValue	UseSiebelLoginPassword
Append as Argument	True (checked)

Testing the Siebel ICM Tab in Siebel CRM

After you have completed the configuration process, make sure the new Siebel CRM tab is in place and functions properly.

This task is a step in "Process of Configuring Siebel CRM to Siebel ICM Single Sign-On" on page 165.

To test the Siebel ICM tab in Siebel CRM

1 Log out from your Siebel CRM application.

2 Log in to your Oracle's Siebel Customer Relationship Management (CRM) product family application again.

A new tab labeled "Siebel ICM" appears.

TIP: If you do not see the "Siebel ICM" tab, make sure that you have added this view to a responsibility that your own user account has been granted. See "Adding Siebel ICM Screen Access to Responsibilities" on page 173.

3 Click the Siebel ICM tab.

A new window opens displaying the Siebel ICM home page.

23 Setting Up Report Documents

This chapter explains how to use and configure reports in the following topics:

- "About Report Documents and Actuate Server" on page 177
- "Process of Setting Up Report Documents with Siebel ICM" on page 177

About Report Documents and Actuate Server

A *report document* is the stored output of an executed report executable. A *report executable* is a compiled report design file that end users can run on demand. Using either the Actuate Management Console or the Actuate e.Report Designer Pro, you can store the output of a report execution as a report document (.roi) file. For more information about report executables, see the chapter on reports in the *Siebel Incentive Compensation Management Administration Guide*.

Instead of using report executables or on-demand reports, there are situations where it is more useful to run report documents or prerun reports. For example, you can:

- Use batch scheduling to generate reports that have low or complex reports with slow execution times.
- Use Actuate's page-level security features to receive a better performance than on-demand reports or to secure access to data. In addition, page-level security can only be used with Report Documents.
- Store a snapshot view of Siebel ICM data for a given amount of time. On-demand reports are transient, so the output of the report is only available at the time the report is run. A report document is a permanent document of the state of the system at the time the report was run.

Process of Setting Up Report Documents with Siebel ICM

To use the report documents feature with Siebel ICM, system administrators must first set up the Actuate Server to use report documents. Administrators must also configure the ways to use report documents within Siebel ICM.

The process of setting up report documents with Siebel ICM is as follows:

1. "Adding an ICM Folder on the Actuate Server" on page 178
2. "Granting the ICM System User Access to the Actuate Folder" on page 178
3. "Defining Report Document Security Access" on page 179
4. "Making Report Documents Available in Siebel ICM" on page 179

Adding an ICM Folder on the Actuate Server

To use report documents in Siebel ICM, you must first create a folder that is visible to Siebel ICM to store report documents. This step requires familiarity with the Actuate Management Console. For more information about the Actuate Management Console, see the Actuate Server documentation.

If you have already loaded report executables into the Actuate Server through Siebel ICM, the folder will already be there with the following name:

`<Siebel ICM Root>/<Operating Unit Code>`

The default value for `<Siebel ICM Root>` is `SiebelICM`, but the value can be modified at installation time.

If this has been modified, see the property reporting.actuate.root in the file reporting.properties to determine the value.

`<Operating Unit Code>` is the name of the operating unit with which you want to use report documents. If you want to use report documents with more than one operating unit, a separate folder needs to be created for each operating unit.

This task is a step in "Process of Setting Up Report Documents with Siebel ICM" on page 177.

To add a folder for ICM Report Documents on the Actuate Server

1. Log in to Actuate Management Console as the volume administrator for the volume being used by Siebel ICM.
2. Siebel ICM uses the following folder to handle report documents:

   `<Siebel ICM Root>/Operating Unit Code>/reportDocuments`

   If the reportDocuments folder does not exist, add this folder.

Granting the ICM System User Access to the Actuate Folder

After the folder is created on the Actuate server, you grant access to this folder to the Siebel ICM System User, typically SiebelICM.

This task is a step in "Process of Setting Up Report Documents with Siebel ICM" on page 177.

To grant access to the Siebel ICM System User

1. View the Properties for the reportDocuments folder created in the previous procedure.
2. Click the Privileges tab.
3. Click the Users button, and grant the Siebel ICM System User the All privilege.
4. Click OK.

Defining Report Document Security Access

Report document security is managed using the Actuate Management Console's security capabilities. For each report document, administrators must define which users have access to that document.

ICM users can only view report documents that they have security access to, as defined using the security privileges on the Actuate Server. Therefore, for your users to view report documents, administrators must first define which users have access to that document.

This task is a step in "Process of Setting Up Report Documents with Siebel ICM" on page 177.

To grant access to report documents

1. Log in to the Actuate Management Console as the volume administrator for the ICM volume.
2. Create an Actuate Server user for each ICM user that you want to have report document access.

   The user name must be identical to the ICM user name. Because of Actuate and ICM security integration, Actuate Server users do not require a password.
3. Give users access to the reportDocuments folder, and to any subfolders and report documents you want to make available to those users, by using the Properties and Privileges screen.

   For more information about creating users or granting users security access, see the administering Actuate iServer System chapter of the Actuate documentation.

Making Report Documents Available in Siebel ICM

After you have set up the Actuate Server, you must make report documents available in Siebel ICM. You can do this by either using a report configuration to view report documents or by using a report document browser.

Either way you choose to make report documents available in Siebel ICM, you must first grant access, using Actuate Server, to users. For more information about granting access, see "Defining Report Document Security Access" on page 179.

The Report Document Browser is one of the Dashboard Area Content Types available when configuring the ICM dashboards. The Report Document Browser area allows users to browse folders on the Actuate Server (under the reportDocuments folder only), and view any report documents stored there. For more information about displaying report document browser, see "Displaying Report Documents on a Dashboard" on page 185.

To make report documents available through report configuration, perform the following procedure.

Before defining a report configuration in Siebel ICM, you must first grant the ICM System User security access to the report document. For more information about granting access to the ICM System User, see "Granting the ICM System User Access to the Actuate Folder" on page 178.

This task is a step in "Process of Setting Up Report Documents with Siebel ICM" on page 177.

To define a report configuration to view report documents

1 In Siebel ICM, create a new report configuration by selecting Report Document from the Report Type drop-down list.

For more information about creating report configurations, see the chapter on reports of the *Siebel Incentive Compensation Management Administration Guide.*

2 In the Report Document field, click Search to browse the reportDocuments folder on the Actuate Server, and then select the report document you want to see.

After the report configuration is created, you can use it in report lists or dashboards.

24 Setting Up Dashboards

This chapter explains how to create, manage, and configure dashboards. It includes the following topics:

- "About Dashboards" on page 181
- "Process of Managing Dashboards" on page 181

About Dashboards

In ICM, a *dashboard* is a screen tab that gives users on-screen access to reports or statements. A dashboard may include one or more *dashboard pages*, which contain the content and are accessed from the main tab screen through hyperlinks. The content area of a dashboard page can have one of several layouts.

Process of Managing Dashboards

System administrators create and set up dashboards, and make them accessible to users. The process of managing dashboards is as follows:

1. "Creating a Dashboard" on page 181
2. "Adding Dashboard Pages" on page 182
3. "Adding Content to a Dashboard Page" on page 183
4. "Granting Access to a Dashboard" on page 186

Creating a Dashboard

Application administrators create different dashboards for different user roles. For example, an administrator can create a SalesRep dashboard that will include a list of reports (such as commission or bonus detail reports) that sales representatives can execute on demand. This topic describes how to create a new dashboard.

This task is a step in "Process of Managing Dashboards" on page 181.

To create a dashboard

1. Click the System tab and navigate to Configure > Dashboards.
2. Click the New Dashboard link.

3 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Priority	Place for this tab in the order in which the tabs appear left to right on the top part of the screen.
Render Order	The order in which the pages are built or rendered by the system. Render Order depends on the order in which the tabs for the pages appear.
Icon URL	Path to an image that will be displayed on the tab for the dashboard. If you do not want a graphic icon on this tab, leave this field blank.

4 Click Save.

Adding Dashboard Pages

After you have created a new dashboard, you add dashboard pages to it.

This task is a step in "Process of Managing Dashboards" on page 181.

To add dashboard pages

1 Click the System tab and navigate to Configure > Dashboards.

2 In the Dashboards Found list, locate the dashboard to which you want to add pages and click its View icon.

3 Click the Add Dashboard Page link.

4 Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Priority	Place for this tab in the order that the tabs appear left to right on the top part of the screen.
Render Order	The order in which the pages are built or rendered by the system. Render Order depends on the order in which the tabs for the pages appear.

5 In the Screen Layout box, select a layout for the dashboard page.

6 Click Save.

7 Repeat Step 3 through Step 6 for each page you want to add to the dashboard.

Adding Content to a Dashboard Page

After you have created dashboard pages, you add content to be displayed on them. You add content separately to each content area. The content can be a Web page, ICM reports, test handlers, or a message display. A report you set up as content can be an interactive one that accepts user-entered parameters to specify the limits of the information a user wants to see. For more information about displaying reports, see "Displaying a Report on a Dashboard" on page 184, "Displaying a List of Reports on a Dashboard" on page 185, and "Displaying Report Documents on a Dashboard" on page 185.

This task is a step in "Process of Managing Dashboards" on page 181.

To add content to a dashboard page

1. Click the System tab and navigate to Configure > Dashboards.
2. In the Dashboards Found list, locate the dashboard for which you want to add content to pages and click its View icon.
3. In the Dashboard Pages list, find the dashboard page to which you want to add content and click its View icon.
4. In the Screen Layout Content list, click the Add Content icon.
5. From the Type drop-down list, select the appropriate content handler to determine the type of content displayed.

   Some types of content that you can select are described in the following table.

Type	Comments
Display a Web Page	Select this option if you want to embed an external Web page inside ICM. This is useful, for example, if you want to display content such as corporate policies from your intranet inside the ICM application. You can also embed Web pages that reference external Web sites.
Message Display	Select this option if you want to display a graphical message window as part of a content area. The message has to be entered in the Message Headline 1, 2, and 3 parameters.
Report Document Browser	Select this option if you want to give users access to a report folder on the Actuate server.
Report List	Select this option if you want to present a list of reports configuration to end users. For more information, see "Displaying a List of Reports on a Dashboard" on page 185.
Single Report	Select this option if you want to present a single report configuration to end users. For more information, see "Displaying a Report on a Dashboard" on page 184.

   The other fields change dynamically according to the Type selection.

6 Complete the necessary fields. Some of the fields are described in the following table.

Field	Comments
Content Area	Select the content area of the layout in which you want this content to appear. Areas are numbered left to right and top to bottom. Only those areas that do not have content already are available for selection.
IFRAME	When selected, navigation inside the embedded content remains within the boundaries of the dashboard page. When not selected, clicking a link from a dashboard page will load the new page in full screen mode, removing the navigation frame on the top.
Message Headline 1 Message Headline 2 Message Headline 3	Parameters of the Message Display content type. Each line displays a message within a graphical text box on the screen.
URL	Parameter of the Display a Web Page content type. Insert the URL of the page you want to display in this content area.

7 Click Save.

8 If this dashboard page has multiple content areas, repeat Step 4 through Step 7 for each content area to which you want to add content.

Displaying a Report on a Dashboard

You can present a single report configuration to end users. The end user clicks a dashboard tab, and ICM displays a single report. Depending on the report, ICM may prompt the end user for parameter values.

To display a single report on a dashboard

1 Click the System tab and navigate to Configure > Dashboards.

2 In the Dashboards Found list, locate the dashboard for which you want to add content to pages and click its View icon.

3 In the Dashboard Pages list, find the dashboard page to which you want to add content and click its View icon.

4 Click the Add Content icon.

5 From the Type drop-down list, select Single Report.

6 In the Report Configuration drop-down list, select the code for the report configuration that you want to display on the dashboard.

7 Click Save.

Displaying a List of Reports on a Dashboard

You can present a list of report configurations to an end user. The end user clicks the tab, and ICM displays a list of reports. The list is fully configurable; you can set it up any way you want. You can specify which reports are displayed and how they are displayed—for example, by defining groupings.

To display a list of reports on a dashboard

1. Click the System tab and navigate to Configure > Dashboards.
2. In the Dashboards Found list, locate the dashboard for which you want to add content to pages and click its View icon.
3. In the Dashboard Pages list, find the dashboard page to which you want to add content and click its View icon.
4. Click the Add Content icon.
5. Complete the necessary fields. Some fields are described in the following table.

Field	Comments
Type	Select Report List.
Content Area	Select the content area of the layout in which you want this content to appear. Areas are numbered left to right and top to bottom. Only those areas that do not have content already are available for selection.
IFRAME	Select the IFRAME option to ensure that the navigation within the reports remain within the context of the dashboard page.
Report List	Select a report list from the list of reports created for your operating unit. Report Lists are created under the Master Control > Report Lists menu. For more information about report lists, see the reports chapter of *Siebel Incentive Compensation Management Administration Guide*.

6. Click Save.

Displaying Report Documents on a Dashboard

You can set up a dashboard tab area that allows end users to browse through the Encyclopedia Volume folders to which they have access and view the pre-run report documents these folders contain. This allows you to make report documents available to end users without creating report configurations for those report documents. You define access to a root folder, and end users see only this root folder and the subfolders below it in the hierarchy.

For more information about the report document folders in the Encyclopedia Volume, see the chapter on reports in *Siebel Incentive Compensation Management Administration Guide*.

You can set up a dashboard tab area that allows end users to browse through the Encyclopedia Volume folders to which they have access and view the pre-run report documents these folders contain.

To display the report document browser on a dashboard

1. Click the System tab and navigate to Configure > Dashboards.
2. In the Dashboards Found list, locate the dashboard for which you want to add content to pages and click its View icon.
3. In the Dashboard Pages list, find the dashboard page to which you want to add content and click its View icon.
4. Click the Add Content icon.
5. From the Type drop-down list, select Report Document Browser.
6. In the Report Documents folder field, specify the folder name on the Actuate Server that you want to make available for browsing. Folders must be below the reportDocuments folder for the operating unit, and only the folder path below the reportDocuments folder needs to be specified.

   For example, if the folder <Siebel ICM Root>/<Operating Unit Code>/reportDocuments/Participants has been created on the Actuate Server, enter Participants in the Report Documents folder field to allow users to browse that folder.

   If the Report Documents folder field is left blank, the root reportDocuments folder for your Operating Unit (<Siebel ICM Root>/<Operating Unit Code>/reportDocuments) will be used by default.
7. Click Save.

   For more information on how the reportDocuments folder is set up on the Actuate Server, see "Adding an ICM Folder on the Actuate Server" on page 178.

Granting Access to a Dashboard

When you create a dashboard, the system automatically creates a dashboard privilege. For end users to be able to access a dashboard, you must add the privilege to their role.

This task is a step in "Process of Managing Dashboards" on page 181.

To grant access to a dashboard

1. Click the System tab.
2. Navigate to Configure > Roles.
3. In the Roles list, find the role to which you want to add access to a dashboard and click its View icon.
4. Click the Edit icon for the Dashboards security group.
5. Select the check box next to the dashboard you want this role to access.

6 Click Save.

25 Removing Deleted Data

This chapter describes how to configure the mechanism that removes ICM data records that are flagged for deletion. This chapter contains the following topics:

- "About the Reaper" on page 189
- "Configuring the Reaper to Remove Deleted Records" on page 189

About the Reaper

When end users run the Purge Period Data service or rerun a service within the same period, that service marks credit and earning records for deletion. It does so by changing the records' Deleted flag values from 0 (undeleted) to 1 (deleted), but leaves the records in the database.

The actual deletion is accomplished by another process. This process is the *Delete Agent*, also known as the *Reaper*. The Reaper is a programmatic agent within ICM that periodically removes credits and earnings that are marked for deletion from the database. The Reaper starts with the service run for the earliest undeleted credit record with Deleted = 1 and works its way forward in time, deleting records for each subsequent service run. Thus, it removes deleted records in chunks, one or more chunks for each service run, rather than all deleted records at once.

When the Reaper is running, parallel threads delete credits and earnings simultaneously. The Reaper makes multiple connections to the database to delete records from the CREDIT and EARNING database tables at the same time.

If you run the Reaper in Debug logging level mode, the system displays messages providing evidence that it is working. It will show a starting message, running messages, and finally an ending summary message. It also shows how quickly the Reaper is deleting records. For more information about logging level modes, see Chapter 27, "Setting Up Debugging and Logging."

TIP: Configuring the CREDIT and EARNING database tables to reside on separate disks, accessed by separate disk controllers, can yield significant performance improvements.

The Reaper can recover from lost JDBC connections. When a service is running, the Reaper either yields or exits, depending on how you set it up.

Configuring the Reaper to Remove Deleted Records

You can configure various aspects of the Reaper's behavior, such as when and how often it runs. To configure the Reaper's deleted records removal behavior, follow this procedure.

To configure the Reaper to remove deleted records

1 Navigate to the following directory:

```
<STAGING>\config\agentConfig\
```

<STAGING> refers to the directory where you configure the ICM files before deployment.

2 Open the agent.cfg.xml file in a text editor and locate the section that begins with the following line:

```
<agent name="Reaper" className="com.siebel.icm.agent.fastDelete.Reaper">
```

This identifies the Reaper agent section of the file's XML code.

3 In the Reaper agent section, set the properties according to the requirements of your configuration.

Each property has the following syntax:

```
<property name="NAME" value="VALUE"/>
```

The properties and their values are listed in the table that follows.

Property	Comments
creditMultiplier earningMultiplier	These settings tell the Reaper to remove deleted credit and earning records, respectively, all at once or in increments. The value must be > 0 and <= 1. ■ To remove all records at once, set the value to 1. A value of 1 attempts to remove all the deleted credits or earnings associated with one service run during one Reaper pass. ■ To remove the records in increments, set the value to a decimal fraction between 0 and 1. In other words, if the value is .5, two Reaper passes will occur. For example, a creditMultiplier value of .1 removes 1/10 of the deleted credits and runs 10 times to remove all the deleted credits for a particular service run. Set the value so the chunk size is as large as possible without running into the limitations of your database resources (in other words, transaction rollback segment space). For more information, see "About the Reaper Transaction Isolation Level" on page 193.
reapNonCreditGenRecords reapNonEarningGenRecords	These settings tell the Reaper whether to limit delete queries for credit and earning records, respectively, to time periods when no service is running. Theoretically, there should be no system-generated records to delete in the times between service runs. However, end users may create and delete records manually during these times. Values are true and false. A value of true removes records manually created and deleted between service runs. A value of false does *not* remove records manually created and deleted between service runs. A value of false can improve the Reaper's performance.
pauseDurationMS	The Reaper does not issue a single delete query to remove all rows that are flagged for deletion. Instead, the Reaper deletes the qualifying records in chunks. The pauseDurationMS value is the amount of time in milliseconds that the Reaper pauses before issuing each subsequent delete statement.

Property	Comments
behaviorWhenServicesAreRunning	While the Reaper runs, it monitors the system for services running. If it detects services running, this property tells the Reaper how to respond. Values are yield and exit. If the value is yield, the Reaper resumes after the services are finished running. If the value is exit, the Reaper stops running and exits.
maxSQLExceptionRetries	This property tells the Reaper how many retry attempts to make in response to a SQLException. This allows the Reaper to recover from dropped JDBC connections due to network failure or unanticipated server downtime.
timeBetweenSQLExceptionRetriesMS	This property specifies the amount of time in milliseconds the Reaper waits before attempting to reconnect to the database after a SQLException.

For examples of property settings, see "Example of Reaper Configuration" on page 193.

4 In the Reaper agent section, set the schedule tags according to the requirements of your configuration.

Each schedule tag line runs the Reaper once each week. By default, the Reaper runs once a day at 3:00 a.m.

CAUTION: Configure the Reaper to run at times when the application server and database server are up and running, but not when you plan to run service processing. Do not configure the Reaper to run while the controller application machine or the database undergo scheduled maintenance such as virus scans, OS backups, database backups, or service patch application.

Each schedule tag has the following syntax:

```
<schedule dayOfWeek="DAY_NUMBER" hour="HOUR_NUMBER" minute="MINUTE_NUMBER"/>
```

The possible values of a schedule tag are listed in the table that follows.

Tag	Values
dayOfWeek	1 through 7, representing the day of the week starting with Sunday = 1.
hour	1 through 24, representing the hour of the day on a 24-hour clock. For p.m. times, add 12. For example, 5 = 5:00 a.m. and 17 = 5:00 p.m.
minute	0 through 59, representing the minute of the hour.

For examples of schedule tag settings, see "Example of Reaper Configuration" on page 193.

5 Save and close the agent.cfg.xml file.

6 Navigate to the <STAGING>/deploy directory and run the following target:

```
ant fast-deploy
```

7 Restart the application server.

About the Reaper Transaction Isolation Level

The Reaper attempts to run its Delete statements under the lowest JDBC transaction isolation level, Connection.TRANSACTION_NONE. The delete operations do not require ACID transaction functionality, so the Reaper uses this isolation level to avoid constraints imposed by the rollback segment or the transaction log size. However, some databases do not support Connection.TRANSACTION_NONE. In these cases, the Reaper polls the database for the supported transaction isolation level it can use to delete records that puts the least system overhead on the database.

If your database does not support the JDBC transaction isolation level of Connection.TRANSACTION_NONE, it is recommended that you increase the database's transaction rollback segment space to be as large as possible. If you cannot change the database settings, then reduce the size of the chunks the Reaper processes by setting its creditMultiplier and earningMultiplier properties to values less than 1.

Example of Reaper Configuration

An example of a Reaper agent section configuration in the agent.cfg.xml file is as follows:

```
<agent name="Reaper" className="com.siebel.icm.agent.fastDelete.Reaper">
   <property name="creditMultiplier" value="1"/>
   <property name="earningMultiplier" value="1"/>
   <property name="reapNonCreditGenRecords" value="false"/>
   <property name="reapNonEarningGenRecords" value="false"/>
   <property name="pauseDurationMS" value="3000"/>
   <property name="behaviorWhenServicesAreRunning" value="yield"/>
   <property name="maxSQLExceptionRetries" value="5"/>
   <property name="timeBetweenSQLExceptionRetriesMS" value="30000"/>

   <schedule dayOfWeek="1" hour="3" minute="0"/>
   <schedule dayOfWeek="2" hour="3" minute="0"/>
   <schedule dayOfWeek="3" hour="3" minute="0"/>
   <schedule dayOfWeek="4" hour="3" minute="0"/>
   <schedule dayOfWeek="5" hour="18" minute="46"/>
   <schedule dayOfWeek="6" hour="3" minute="0"/>
   <schedule dayOfWeek="7" hour="3" minute="0"/>
</agent>
```

26 Removing Obsolete Objects with JVM

This chapter describes how to configure the JVM mechanism that removes obsolete generic objects. This chapter contains the following topics:

- "About the JVM Garbage Collector" on page 195
- "Configuring ICM to Optimize Multi-Processor Sun JVM Garbage Collection" on page 196

About the JVM Garbage Collector

The Java Virtual Machine (JVM) has a *garbage collector* function, which identifies objects that are no longer referenced by other objects (in other words, objects that are obsolete). It uses a mark and sweep algorithm to reclaim the memory used by those objects.

Sun JVM Garbage Collection Options

You can configure ICM so that it works correctly with the Sun JVM garbage collector function in a multi-processor environment. You do so by changing the garbage collector option. Sun JVM garbage collection options that are suitable for ICM include the following:

- **One or two processors.** When running ICM with the Sun JVM in an environment with one or two processors, it is recommended that you use either the Concurrent Low Pause Collector or the Hot Spot Collector for maximum performance and stability.

  The default ICM option for the Sun JVM is the *Concurrent Low Pause Collector*, a generational collector similar to Sun JVM's default Hot Spot Collector. The Concurrent Low Pause Collector collects the *tenured generation* concurrently. (Tenured generation refers to objects that have already survived several mark-and-sweeps and have moved to longer-term storage.)

  The Concurrent Low Pause Collector attempts to reduce the pause times needed to collect the tenured generation. It uses a separate garbage collector thread to do parts of the major collection concurrently with the applications threads.

- **Three or more processors.** When running ICM with the Sun JVM in an environment with three or more processors, it is recommended that you use the Throughput Collector for maximum stability over long-running processes.

  The *Throughput Collector* is a generational collector that is also similar to Sun JVM's default Hot Spot Collector, except that it uses multiple threads to do the minor collection. (The major collections are essentially the same as those of the default collector.) On a host with *N* CPUs, the Throughput Collector uses *N* garbage collector threads.

  For information on setting up ICM to use the Throughput Collector, see "Configuring ICM to Optimize Multi-Processor Sun JVM Garbage Collection" on page 196.

  CAUTION: If you do not configure ICM to use the Throughput Collector in environments of three or more processors, the JVM may encounter errors during garbage collection, and processing may stop before collection is completed.

For more information about Sun JVM garbage collection, see the Sun documentation at http://java.sun.com/docs/hotspot/gc1.4.2/.

IBM JVM Garbage Collection Options

Although similar in concept to Sun JVM, the IBM JVM and hardware memory management processes differ in details. No additional configuration of ICM is required for IBM JVM garbage collection.

For information about IBM JVM garbage collection, see the IBM documentation at http://www-128.ibm.com/developerworks/ibm/library/i-gctroub/.

Configuring ICM to Optimize Multi-Processor Sun JVM Garbage Collection

You can configure ICM's JVM options to optimize the garbage collection function in Sun JVM environments of three or more processors.

To configure ICM to optimize multi-processor Sun JVM garbage collection

1. Open a command prompt window and navigate to the following directory:

   `<STAGING>/etc/jboss3.0/bin/`

2. Launch a text editor and open one of the following startup files:
   - For UNIX, open the run.sh.in file.
   - For Windows, open the run.bat.in file.
3. In the startup file, locate the following Java options settings line:

   ```
   set JAVA_OPTS=%JAVA_OPTS% -server -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled -XX:+UseConcMarkSweepGC -Dprogram.name=%PROGNAME%
   ```

   This line specifies the Concurrent Low Pause Collector.
4. Replace the Java options settings line with the following line:

```
set JAVA_OPTS=%JAVA_OPTS% -server -XX:+UseParallelGC -XX:ParallelGCThreads=<CPU
Count> -XX:+UseAdaptiveSizePolicy -Dprogram.name=%PROGNAME%
```

This new line specifies the Throughput Collector. For <CPU Count>, substitute the number of CPUs on the system that is running the JVM. This number tells the Throughput Collector how many garbage collection threads to use.

5. Save and close the startup file.
6. Shut down the application server.
7. Navigate to the <STAGING>/deploy directory and run the following target:

   ```
   ant fast-deploy
   ```

8. Restart the application server.

27 Setting Up Debugging and Logging

This chapter describes how to configure ICM to generate and record debugging information, which Oracle Technical Support uses to help you fix errors. It includes the following topics:

- "About Debugging and Log Files" on page 199
- "Process of Generating Debugging Information" on page 200
- "About Appenders and Categories" on page 202

About Debugging and Log Files

While configuring ICM, system administrators and IT professionals may get error messages that they do not understand or that they cannot solve on their own. The ability to track errors and processes in the application is provided by log files. Several different types of log files are provided with the application. You can adjust settings for the logs as required.

ICM uses Jakarta Log4j to control logging. ICM provides the following log4j configurations:

- **Release.** This configuration is enabled by default and logs only serious errors.
- **Debug.** This configuration is very verbose and is not suitable for large deployments.

For more information, see the Jakarta Log4j documentation.

Types of ICM Log Files

This topic describes the various types of log files present in Siebel ICM.

Application Log

The Application Log tracks normal application use. Tasks tracked include logging on to the application, selecting entities from the main menus, creating new entities, and so on.

All logging for the Application Log file is controlled by log4j. The log configuration files are in the <STAGING>/config directory. To make changes to the logging in release mode, edit the log4j.release.in file. To make changes to the logging in debug mode, edit the log4j.debug.in file.

NOTE: <STAGING> refers to the directory where you configure the ICM files before deployment.

Boot Log

The Boot Log contains information about the startup sequence of the application server during application startup. The application server generates this file during startup. For JBoss, this log file is in the <DEPLOYMENT>/jboss-3.0.3_tomcat-4.1.12/server/default/log directory and the output file name is boot.log. For WebSphere, this log file is in the <DEPLOYMENT>/IBM/WebSphere/AppServer/logs/server1 directory and the output file name is startServer.log.

NOTE: <DEPLOYMENT> refers to the root directory of your ICM installation.

You cannot change content generated by the application server. Consequently, you cannot adjust the settings for this log file.

LocalHost Access Log

The LocalHost Access Log contains information about the activity of the Web server. For JBoss, this log file is in the <DEPLOYMENT>/jboss-3.0.3_tomcat-4.1.12/server/default/log directory. For WebSphere, this log file is in the <DEPLOYMENT>/IBM/WebSphere/AppServer/logs/server1 directory.

The system appends the current server date and time to this log file name. An example of a JBoss LocalHost Access log name is localhost_access2005-03-09.log. An example of a WebSphere LocalHost Access log name is SystemOut_05.03.09_14.35.53.log.

You cannot adjust the settings for this log file.

Process of Generating Debugging Information

You adjust log levels up (increased information) when you have a problem with the application. After solving the problem, you adjust them back down.

The tasks for generating debugging information are as follows. Do them in the order listed.

NOTE: To use these procedures, you must have access to the file system on the computer that has ICM installed.

1. Increase the logging detail to the maximum (Debug) configuration. See “Changing the Logging Configuration” on page 201.
2. If necessary, further adjust the logging detail. See “Adjusting the Logging Level in the Application Log File” on page 201.
3. Generate and send the logs to Oracle Technical Support. See “Generating and Sending the Logs” on page 202.
4. Use the input from Oracle Technical Support to solve the problem.
5. Reduce the logging detail to the normal (Release) configuration. See “Changing the Logging Configuration” on page 201.
6. If necessary, further readjust the logging detail. See “Adjusting the Logging Level in the Application Log File” on page 201.

Changing the Logging Configuration

Use this procedure to turn on maximum logging detail, which is useful for debugging, or to restore minimal logging detail, which is sufficient for normal ICM use.

This task is a step in "Process of Generating Debugging Information" on page 200.

To change the logging configuration

1. Navigate to the <STAGING>/deploy/ directory.
2. Open the deploy.default.properties file in a text editor and make one of the following changes to the logging mode property value:
   - To change to Debug mode, which gives a greater level of detail in logging information:

     `deploy.logging.mode=debug`
   - To change to Release mode, which gives a lesser, normal level of detail in logging information:

     `deploy.logging.mode=release`
3. Save the deploy.default.properties file and close it.
4. In the <STAGING>/deploy directory, run the following target:

   `ant deploy-logging`
5. (Optional) Restart the application server.

   The log4j settings will take effect without a reboot. The line number detail in stack traces will only change with a reboot.

Adjusting the Logging Level in the Application Log File

After you have selected release mode or debug mode, use this procedure to further adjust the logging level in the Application Log file.

NOTE: This topic describes making changes to the log4j files directly. Those changes are overwritten if you run a deploy target; for example, deploy, fast-deploy, or deploy-logging.

This task is a step in "Process of Generating Debugging Information" on page 200.

To adjust the logging level in the Application Log file

1. Navigate to the <STAGING>/config directory.
2. In the directory, do one of the following to adjust the logging level:
   - In release mode, open the log4j.release.in file.
   - In debug mode, open the log4j.debug.in file.

3 Change the appender definitions as necessary. See "Appender Definitions for ICM Log Files" on page 202.

CAUTION: Only advanced users should change the appenders.

4 Change the category definitions as necessary. See "Category Definitions for ICM Log Files" on page 205.

5 Navigate to the <STAGING>/deploy directory and run the following command:

```
ant deploy-logging
```

Generating and Sending the Logs

After you have increased the logging level detail, use this procedure to generate error logs and send them to Oracle technical support.

This task is a step in "Process of Generating Debugging Information" on page 200.

To generate and send the logs

1 Reproduce the error.

This generates the detailed logs.

2 In the <STAGING>/deploy directory, run the following target:

```
ant backup-logs
```

3 In the <STAGING>/backup/logs directory, locate the logs' ZIP file.

4 Email the logs' ZIP file to your Oracle technical support representative.

About Appenders and Categories

Each log file configuration consists of an *appender*, which directs where the output should go, and a *category*, which specifies what types of messages are logged. You can make changes to logging levels in the category declaration.

The following topics provide detailed information about appenders and categories:

- "Appender Definitions for ICM Log Files" on page 202
- "Category Definitions for ICM Log Files" on page 205

Appender Definitions for ICM Log Files

You can configure the appenders to change the name and location of log files, message formats, file size maximums, and maximum number of files to keep for historical purposes.

Console Appender

If the server is running in standalone, non-service mode, this appender sends logged output to the application server startup window.

```
<appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
     <param name="Target" value="System.out"/>
     <param name="Threshold" value="WARN"/>
     <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>
```

Advanced Rolling File Appender

This appender separates infrastructure debugging and logging information from application-specific debugging and logging information. For JBoss, the advanced rolling file appender logs to a file in the <DEPLOYMENT>/jboss-3.0.3_tomcat-4.1.12/bin directory. For WebSphere, the advanced rolling file appender logs data to a file in the <DEPLOYMENT>/IBM/WebSphere/AppServer directory.

```
<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="Threshold" value="ERROR"/>
     <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>
```

Application Rolling File Appender

This appender logs application-level, not application server-level, debugging and logging information to a file that the system splits into individual logs of 4096 KB each. This file appears in the same directory as the Siebel_Advanced_Log.html file.

```
<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>
```

Application Server Rolling File Appender

This appender sends application server (j2ee) logging messages to a server.log file. This can help diagnose application server infrastructure issues.

- For JBoss:

```
<appender name="APPSERVER"class="org.apache.logging.appender.DailyRollingFileAppender">
   <param name="File" value="<DEPLOYMENT>/server/default/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>
```

- For WebSphere:

```
<appender name="APPSERVER"class="org.apache.logging.appender.DailyRollingFileAppender">
   <param name="File" value="<DEPLOYMENT>/IBM/WebSphere/AppServer/logs/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>
```

For more information about logging messages, see “Filtering Security-Related Log Messages” on page 125.

Audit Log Appender

This appender sends auditing information to a file for manual inspection. Audit events exist for the following items:

- Earning adjustments
- Role creation
- Role update
- User creation
- User update

```
<appender name="AUDIT" class="org.apache.log4j.RollingFileAppender">
     <!-- pick your output file name -->
     <param name="File" value="Siebel_Audit_Log.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>
```

Category Definitions for ICM Log Files

To change the level of logging for a given category, you must change the category definition. Each category definition can consist of the following items:

- **Name.** Name of the category. Do not change this, because Siebel ICM sends log messages to this name internally.
- **Priority.** Threshold that must be met or exceeded for the log message to be sent to the appender. The logging level priority settings available for the log files are listed in ascending level of detail in Table 19.

Table 19. Logging Level Priority Settings

Logging Level	Comments
Fatal	Provides severe error messages and processing times.
Error	Provides all error messages and processing times. Use this setting when reporting issues to Oracle 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 Oracle Technical Support.

The more detailed priority settings (Debug, Info, and Warn) can create very large files and can impact the overall performance of the application. Use these settings only as directed by Siebel Professional Services. Additionally, Console logging has a much greater impact on performance than File logging. It is recommended that you turn off all Console logging in a production environment.

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

For more information about logging categories, see “Filtering Security-Related Log Messages” on page 125.

Examples of category definitions are as follows:

com.motiva.dashboards

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

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

Audit

Messages logged to the Audit category go to the Audit appender. Only messages of level INFO or above are logged.

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

MotivaSystemMessage

System-level messages logged to the MotivaSystemMessage category go to the Rolling File appender. Only messages of level DEBUG or above are logged.

```
<category name="MotivaSystemMessage">
<priority value="DEBUG" />
<appender-ref ref="RFA"/>
</category>
```

Leave this category's priority value set to DEBUG. This category is used to log the startup message, for example:

```
10:37:35,389 [SiebelICMSystemMessage] ****************************************
10:37:35,399 [SiebelICMSystemMessage] ****************************************
10:37:35,399 [SiebelICMSystemMessage]
10:37:35,399 [SiebelICMSystemMessage] Siebel ICM
10:37:35,399 [SiebelICMSystemMessage] Version Drago
10:37:35,399 [SiebelICMSystemMessage] Build [325]
10:37:35,399 [SiebelICMSystemMessage] Started Successfully at 10:37:35
10:37:35,399 [SiebelICMSystemMessage] Started in location: 'd:\jboss_tomcat_main\bin'
10:37:35,399 [SiebelICMSystemMessage]
10:37:35,399 [SiebelICMSystemMessage] ****************************************
10:37:35,399 [SiebelICMSystemMessage] ****************************************
```

developer

Developer (Advanced) log messages go to the Advanced Rolling File Appender. All messages of level WARN or higher are logged.

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

com.motiva

General debugging messages for the Siebel ICM platform go to this category. Messages of ERROR or higher are logged.

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

FrameWork

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

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

org.jboss.jdo.castor

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>
```

org.jboss

Application server messages generated by the JBoss application server are logged to the Application Server Log Appender. Only messages of INFO or higher are logged.

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

org.apache

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

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

root

This is the “catch-all” for log messages. It should not be edited.

```
<root>
      <appender-ref ref="CONSOLE"/>
      <appender-ref ref="APPSERVER"/>
</root>
```

A Smart Attribute Scripts Reference

This appendix displays a sample smart attribute script and includes the following topics:

- "About Smart Attribute Scripts" on page 209
- "Example Smart Attribute Script" on page 209

About Smart Attribute Scripts

The smart attribute script that follows is part of standard ICM installations. The explanation of the script can help you generate your own customized scripts for additional smart attributes you create.

NOTE: The other standard smart attribute scripts in ICM follow the same pattern as the one outlined in the topic that follows.

Example Smart Attribute Script

You can use the Customer Code smart attribute in distribution 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
//
// Smart Attribute: Customer Code of Sales Transaction
```

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

```
importClass(Packages.com.motiva.framework.logging.LogService);
importClass(Packages.com.motiva.framework.jndi.JNDIService);
var CAT = "com.motiva.install.script.CaCustomerCode.script";
```

This script block is a standard part of all prepackaged scripts in ICM. 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 it the value found in the Customer field on a transaction header.

NOTE: This line is run in the context of a line and an event on that line because these are always known during crediting. Though the customer code is located on the transaction header, all transaction lines 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 no customer is listed on the transaction header. If so, the system sends a message to the logging service that the customer code could not be found, and the script does not return a value for the customer code. Consequently, any distribution rule credit that tries to write the customer code to a credit record leaves that field blank. This will cause problems only if that field is required.

```
// 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 writes to a credit record that references this smart attribute.

B Template Scripts Reference

This appendix displays sample template scripts and includes the following topics:

- "About Template Scripts" on page 211
- "Example Condition Template Scripts" on page 211
- "Example Recipient Template Scripts" on page 216

About Template Scripts

A *template* in ICM is a script function written in JavaScript that builds either a selection condition or a credit recipient group. A *condition template* builds the selection conditions in distribution rules and plan rules. A *recipient template* selects the participants and organizations that receive distribution rule credits.

The templates described in this appendix are part of every standard ICM installation. The explanations of the JavaScript codes may help you construct your own customized template scripts.

NOTE: In the template topics of this appendix, a double forward slash (//) indicates a comment line, which the system ignores.

Example Condition Template Scripts

The following topics show sample condition template scripts:

- "Line Product Is Specified Product" on page 211
- "Line Field (String) <operator> <value>" on page 213
- "Hdr Field (String) <operator> <value>" on page 215

Line Product Is Specified Product

This condition template gets the value in the Product Code field on a transaction header and compares it to a value specified by an end user. The following JavaScript code handles this function:

```
// AssrCreditProdIsProd.script
//
// Crediting Assessor Template: Is Transaction Line Product equal to Specified
product
//   Parameters:
//      - productUUID
```

These comment lines tell you the script's name, its function, and the parameters it uses.

```
importClass(Packages.com.motiva.framework.logging.LogService);
importClass(Packages.com.motiva.framework.jndi.JNDIService);
var CAT="com.motiva.ce.install.script.AssrCreditProdIsProd.script";
LogService.info(CAT,"Credit Assessor Template: AssrCreditProdIsProd");
```

This block is a standard part of all prepackaged ICM scripts. The first two lines call services included in every script. The next line creates the variable CAT and assigns the script name and location to it. 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 ICM and the unique ID codes (the UUIDs) that ICM automatically assigns to those entity codes. Normally, these *hash codes* are hidden from users and cannot 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 creates the variable `productUUID` and assigns it the text value that the user entered when the condition template was used in a distribution rule. This value changes from one use of the condition template to another. The third line logs this value and labels it for 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: 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, when processing transactions through distribution rules, ICM automatically keeps track of what line and which event on that line it is processing.

Though retrieved in the context of a line, the product code is recorded on a transaction's header. This is valid because the transaction's lines inherit all the header data except for participants.

```
if ( lineProduct == null )
{
  LogService.info(CAT,"No Product defined for Transaction Line in
AssrCreditProdIsProd.script.");
  return(false); }
```

This block addresses transactions that do not reference a product. The system writes a message in the log file that the crediting process failed because no product was found on the transaction. The system also returns a value of false for the script. In this case, the remaining conditional blocks do not run.

```
if ( !(lineProduct.getUUID().equals(productUUID)) )
{
  LogService.info(CAT,"Transaction Line Product does not match Product for Rule in
AssrCreditProdIsProd.script.");
  return(false); }
```

This block catches all the transaction lines for which the transaction's product code does not match the product code set up for the distribution 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 means that the system found a product code on the transaction that met the requirements of the condition. If this template is the only one used in a distribution rule, then the transaction line meets all the rule's conditions and the system creates distributions for the transaction line.

Line Field (String) <operator> <value>

This condition template gets data from a profile attribute on a transaction line, then compares that value to a value a user specifies for the distribution rule. The template accomplishes the comparison with a conditional operator that the user also specifies. For example, a user might want the distribution rule to check whether the Sales Amount on a transaction line exceeds a certain value, such as 30,000. In this case, the template as used in the distribution rule would appear as follows:

```
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.motiva.framework.logging.LogService);
importClass(Packages.com.motiva.framework.jndi.JNDIService);
importClass(Packages.com.motiva.ce.util.script.Evaluator);

var CAT = "com.motiva.ce.install.script.AssrCreditLineReadingCompare.script";
LogService.info(CAT,"Assessor Template: AssrCreditLineReadingCompare");
```

These opening lines of the script are similar to the opening lines of the Line Product Is Specified Product script. For descriptions of the lines, see "Line Product Is Specified Product" on page 211.

NOTE: This script also calls the Evaluator, a JavaScript utility that compares one value to another according to various comparison operators. The Evaluator will be referenced again near the end of the script.

```
// 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 declares three variables, each corresponding to one of the three input values that a user must specify when setting up the template within a distribution rule.

NOTE: These values are automatically generated when you set up the initial parameters of the condition template, but the system assigns them generic names. The names used in this script (readingTypeUUID, operator, stringValue) are manually modified to make the variables more easily identifiable elsewhere in the script and their functions more readily apparent.

The last three lines of this block log the values that the user specified and labels them so that anyone reading the log report can identify the parameters.

```
// look up the reading type
var readingTypeServicer = ServicerBag.instance().getReadingTypeServicer();
var readingType = readingTypeServicer.get(readingTypeUUID);
if (readingType == null)
{

  LogService.error(CAT, "ReadingType not found for readingTypeUUID->" +
readingTypeUUID);
  return(false);}
```

This block checks that the profile attribute the user referenced when setting up the distribution rule exists in the system. The first half of the block is a single function broken into three variable declarations. It could have been written as a single variable declaration, as follows:

```
var readingType =
ServicerBag.instance().getReadingTypeServicer().get(readingTypeUUID);
```

Breaking this function into three separate variables makes the script easier to understand while accomplishing the same task with no loss of speed or functionality.

This function gets the profile attribute ID that the user specified for the condition and searches for that profile attribute ID in the database. The second half of the block addresses the contingency that the user entered a profile attribute ID that does not exist. In this case, the system notes in the log file that the condition failed because the specified profile attribute does not exist, then returns 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 block assumes that the previous conditional block did not run; in other words, the user specified a valid profile attribute. The first line creates a new variable, lineReading, and assigns it the value of the selected profile attribute for the current transaction line. If there is no value in this profile attribute, the system notes in the log file that the distribution rule failed because the transaction line had no value for this profile attribute, and returns a value of false for the condition.

```
var lineStringValue = lineReading.getStringValue();
var retVal = Evaluator.strCompare(lineStringValue, operator, stringValue);
return(retVal);
```

The final block assumes that a valid string value exists in the selected profile attribute. The system gets this 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 compares the value in the selected profile attribute with the value that the user set up for the condition. The Evaluator then uses the operator variable to determine how to make the comparison.

For example, if the user selected Is Greater Than for the operator, then the Evaluator checks whether the profile attribute value on the transaction line is larger than the value the user selected for the condition.

Finally, the value generated for the retVal variable (true or false) is returned as the final result of the condition.

Hdr Field (String) <operator> <value>

When using the IS LIKE operator, the AssrCreditHdrReadingStrComp condition template works with the <value> entries listed in the table that follows:

Entry	Comments
^<value>.*	<value> starts the string.
.*<value>.*	<value> is contained within the string.

For example, consider a header reading called Color. For three transactions, the values for this header reading are Blue, Red, and Blue-green, respectively.

Suppose the condition (^<value>.*) in the credit rule is as follows:

```
If Header Field Color is like ^Blue.*
```

In this case, transactions with header reading values of Blue and Blue-green qualify for this condition, and the system generates credits only for those transactions.

Now suppose the condition (.*<value>.*) in the credit rule is as follows:

```
If Header Field Color is like .*e.*
```

In this case, all header readings containing an “e” match the condition. All three header reading values qualify for this condition, and the system generates credits for all three transactions.

Example Recipient Template Scripts

The following topics show sample recipient template scripts:

- “Line Rep of Given Rank” on page 216
- “Line Rep of Given Rank’s Organization at Given Level” on page 218

Line Rep of Given Rank

This recipient template scans the line rep participants listed on a transaction line and selects the one assigned to a rank that the user specifies when setting up the distribution rule credit. For example, if the user specifies a rank of 3, then the system selects any participant assigned to rank 3.

```
// RecipLineRepByRank.script
//
// Crediting Recipient Template: Return Credit Recipient Collection containing
indicated Line Rep
//   - Parameters:
//      - Rank
//
```

These comment lines tell you the script’s name, its function, and the parameters it uses.

```
importClass(Packages.com.motiva.framework.logging.LogService);
importClass(Packages.com.motiva.framework.jndi.JNDIService);
var CAT = "com.motiva.ce.install.script.RecipLineRepByRank.script";
LogService.info(CAT,"Recipient Template: RecipLineRepByRank");
```

This block is a standard part of all prepackaged ICM scripts. The first two lines call services included in every script. The next line creates the variable CAT and assigns the script name and location to it. 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 ICM and the unique ID codes (the UUIDs) that ICM automatically assigns to those entity codes. Normally, these *hash codes* are hidden from users and cannot 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 it the value that a user enters when the recipient template is used in a distribution rule. This value changes from one use of the template to another, according to the rank to which the user wants to assign credits. The third line logs this value and labels it for ease of reference.

```
// Output Collection
var recipients = new java.util.ArrayList();
```

This line returns a list of participants or organizations that will receive credit distributions. This list is handled in JavaScript by creating the variable recipients, an array of values. 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.

```
// 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 that identifies the line rep with the rank selected for the distribution 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 checks the first line rep's rank on the transaction line. If it matches the rank selected for the distribution rule, it writes that participant's ID to the participant variable. The looping condition stops and the rest of the script proceeds. If the participant's rank does not match the rank selected for the distribution rule, the loop goes back to the transaction line, gets the next line rep, and repeats the process. This continues until the system either matches a line rep to the condition, or runs 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);
}
```

If no line rep could be found to match the rank selected for the distribution rule, then the system notes this fact in the log file and returns an empty list of participants. In this case, no one receives credits according to the distribution.

```
// Add Participant to Collection

recipients.add(participant)

return java.util.Collection(recipients);
```

The last lines of the script run if the script finds a valid line rep of the specified 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 recipient template finds a line rep of a specified 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 find the correct organization at the proper level, the script refers to both the transaction line and the organization hierarchy.

```
// 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.motiva.framework.logging.LogService);
importClass(Packages.com.motiva.framework.jndi.JNDIService);
importClass(Packages.com.motiva.ce.bean.ServicerBag);

var CAT = "com.motiva.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.

These opening lines of the script are similar to the opening lines of the Line Rep of Given Rank script. For descriptions of the lines, see "Line Rep of Given Rank" on page 216.

```
// 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 declares two variables, which correspond to the specified line rep rank and the selected organization hierarchy level. The user must enter these variables when setting up a distribution.

NOTE: These values are automatically generated when you set up the initial parameters of the condition template, but the system assigns them generic names. The names used in this script (rank and orgLevelUUID) are manually modified to make the variables more easily identifiable elsewhere in the script and their functions more readily apparent. The script also writes the user's specified values for each variable to the log file.

```
// Output Collection
var recipients = new java.util.ArrayList();
```

This line returns a list of participants or organizations that will receive credit distributions. This list is handled in JavaScript by creating the variable recipients, an array of values. 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.

```
// 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 java.util.Collection(recipients);
}
```

This block finds the line rep with the specified rank. If it cannot find a line rep with the correct rank, it updates the log service to reflect this fact and the script ends, returning a blank list of recipients.

```
// Get Employee Servicer
var employeeServicer = ServicerBag.instance().getEmployeeServicer();

// 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 java.util.Collection(recipients);
}
```

After the script has found a valid line rep and the records that line rep's participant ID, this block checks whether that participant is in an active job. If it cannot find a current job for the employee, then the employee cannot report to an organization. In this case, there will be no recipients for this distribution.

The first part of this block gets the Employee Servicer, a standard tool in ICM. The second part of the block runs this tool and checks the employee's records to find an active job record for the employee. If no active job is found, then the system logs this fact in the log file and stops the script.

```
// 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 java.util.Collection(recipients);
}
```

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 hierarchyServicer = ServicerBag.instance().getHierarchyServicer();
```

This block refers to the Hierarchy Servicer, which will perform two duties in the remainder of the script.

```
// 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);;
}
```

This block directs the Hierarchy Servicer to perform its first job (getHierarchyLevel). It gets the hierarchy level that the user specified for the distribution and determines whether 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();
var hierarchy = hierarchyServicer.getOrganizationDefaultHierarchy();
orgAtLevel =
hierarchyServicer.getRelatedHierarchyMemberAtLevel(hierarchy.getUUID(),
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);
}
```

This block directs the Hierarchy Servicer to perform its second job. First, the script creates two new variables that the Hierarchy Servicer will refer to later. The orgAtLevel variable holds the organization that the servicer finds at the appropriate hierarchy level. The auditDate variable holds the current date. Then, the Hierarchy Servicer searches for an organization at the hierarchy level the user specified for the distribution.

The Hierarchy Servicer will find a valid organization if, at any time during the current processing period, an organization was at the specified hierarchy level. This allows the organization hierarchy to change during a period, while still allowing the distribution rule to distribute credits to the correct organization.

If the system cannot find an organization in the specified level for the current period, then it 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
recipients.add(orgAtLevel);

return java.util.Collection(recipients);
```

If the script reaches this point without closing (that is, if all conditions have been met), this block adds the organization ID found through the previous steps to the recipients array. This organization becomes the sole recipient for this distribution.

C Informatica Transaction-Loading Mappings Reference

This appendix describes the reference (example) implementation of a group of Informatica Mappings for loading sales transactions directly into Siebel ICM's transactional database tables. It contains the following topics:

- "About the Transaction Model" on page 223
- "Process of Mapping the Sales Transaction Loader" on page 224
- "About Source Data File Formats" on page 225
- "Informatica Objects for the ICM Transaction Data Model" on page 230
- "ICM Sales Transaction Data Model" on page 232
- "Transaction Loader Limitations" on page 233
- "Best Practice for Informatica Analytics Updates" on page 234

About the Transaction Model

This topic describes the *transaction model* for which the reference implementation is designed. Transaction model refers to the data model for the transaction data; in other words, how a sales transaction is modeled within the transaction database.

Transaction Header (Item). This object has the following characteristics:

- Is one record
- Has the following profile attributes:
  - Total Sales Amount (currency)
  - Total Quantity Sold (number)
  - Promotion Code (string)
- Has up to three sales representatives
- Can have any number of Transaction Line records

Transaction Line. This object has the following characteristics:

- Has one event
- Is one record, a child of the Transaction Header
- Has the following profile attributes:
  - Sales Amount (currency)
  - Quantity Sold (number)
  - Expected Ship Date (date)

- Discount Code (string)
- Discount Flag (Boolean)

- Has up to three sales representatives

The profile attributes must already be in ICM for the transactions to load. If you change the profile attribute name, you must make corresponding changes to the Informatica mappings. The data files must have these profile attributes in the order shown in the file formats in "About Source Data File Formats" on page 225.

The data types of the profile attributes are significant, because the mapping must convert or transform the source data into a format that is valid for ICM.

You must make the mapping parameters and variables to correspond to your operating unit code for each mapping. For more information, see "Mapping Parameters" on page 232.

Universally Unique Identifiers

This appendix sometimes refers to *UUID* values. A UUID (Universally Unique Identifier) is an identifier standard used by ICM that allows an application to uniquely identify data items without central coordination. Thus, a user or an automatic process can create a UUID to identify a programmatic object without the risk that another user or process will inadvertently label some other object with that same identifier. Information with UUIDs can therefore be written to the ICM database with no need to resolve name conflicts.

Process of Mapping the Sales Transaction Loader

The sequence of steps in the Sales Transaction Loader Informatica Mapping, referred to from now on as a *mapping*, is as follows:

1. Create two source files for transactions, one file for Header Transactions and the other for Line Item Transactions.

   For information about the formats of these source files, see "About Source Data File Formats" on page 225.

2. Run the Informatica workflows in the correct order. First run wf_SALESTRANS, then run wf_SALESTRANS_LINE.

   These workflows read data from the source files you created in Step 1 and insert transaction data into ICM tables. For more information about the Informatica workflows, see "Workflow, Session, and Mappings" on page 231.

3. Validate the transaction data in ICM by doing the following:

   a. Go to the Transaction database and check the Salestrans table, making sure the records have been transferred.

   b. Launch your ICM application.

   c. Navigate to the Transactions > Transactions view.

d Enter search criteria to identify the transactions you want to validate, and then click Search.

e In the Transactions Found list, find the transactions you want to validate, and then click each one's View icon to open it and view details.

4 Look for any Informatica errors.

a Check the Informatica logs such as pmserver.log, Informatica workflow logs, and Informatica session logs.

b Check for non-empty *bad files*.

A bad file is a file the Informatica Server generates when the transferred data contains an error. Bad files are located in the following directory:

`<INFA_INSTALL>/Server/BadFiles`

For `<INFA_INSTALL>`, substitute the name of the root directory for your Informatica installation; for example, D:\informatica711\Server\BadFiles.

If a file size is 0 bytes, the transferred data has no errors. If a file size is more than 0 bytes, open it to find out what is wrong with the data.

5 Fix the errors and rerun the workflows.

To avoid some common errors, observe the following guidelines:

- Configure the Informatica server carefully. If you do not, errors like "Cannot connect to the repository" might occur.
- In ICM source data file formats, most strings are 20 to 30 characters long. If your data string is longer, change the port position in the mapping to match the string length, or match the string length to the ICM limit. If you do not, the system will truncate the lengthy string, which can cause data errors. For more information, see "About Source Data File Formats" on page 225.

About Source Data File Formats

Two source data files are needed to load transactions into ICM. One file is for Header-level transactions and the other is for Line-level transactions. In both files, Header and Line records are connected by Transaction Number fields. These data files are comma-separated text files. The files and their formats are described in the topics that follow.

Transaction Header File

The transaction header file name is dataset1_header_flat.csv, as used in Informatica mappings. You can change the name as needed. It is a comma-separated text file with field names in the first row. These fields are described in Table 20.

Table 20. Transaction Header File Fields

No.	Field Name	Data Type and Size	NULL Allowed?	Mapped field in ICM	Comments
1	Transaction Number	String(20)	No	SALESTRANS. SALESTRANS_NUMBER	Must be unique for *all* transaction headers for an Operating Unit. Connects header records to line records.
2	Header Date	Date in MM/DD/YYYY format	No	SALESTRANS. HEADER_DATE	
3	Transaction Type	String(30)	No	SALESTRANS. SALESTRANS_TYPE_UUID	UUID calculated from SALESTRANS_TYPE
4	ChannelSegment Code	String(30)	Yes	SALESTRANS.channel_segment_uuid	UUID calculated from channelSegment
5	Customer Code	String(30)	Yes	SALESTRANS.customer_uuid	UUID calculated from customer
6	Territory Code	String(30)	Yes	SALESTRANS.territory_uuid	UUID calculated from territory
7	ShipTo Location	String(30)	Yes	SALESTRANS.ship_to_location_uuid	UUID calculated from location
8	SoldTo Location	String(30)	Yes	SALESTRANS.sold_to_location_uuid	UUID calculated from location
9	Currency Code	String(3)	No	SALESTRANS. CURRENCY_UUID	UUID calculated from CURRENCY
10	Header Description	String(90)	Yes	SALESTRANS. DESCRIPTION	
11	Total Sales Amount	String(20)	Yes	SALESTRANS_READING. ATTRIBUTE_VALUE_NUMBER	Profile Attribute must exist in ICM.

Table 20. Transaction Header File Fields

No.	Field Name	Data Type and Size	NULL Allowed?	Mapped field in ICM	Comments
12	Total Quantity Sold	String(20)	Yes	SALESTRANS_READING. ATTRIBUTE_VALUE_NUMBER	Profile Attribute must exist in ICM.
13	Promotion Code	String(30)	Yes	SALESTRANS_READING. ATTRIBUTE_VALUE_STRING	Profile Attribute must exist in ICM.
14	Sales Rep 1	String(20)	Yes	SALESTRANS_SALESREP. SALESTRANS_SALESREP_UUID	UUID calculated from PARTICIPANT
15	Sales Rep 2	String(20)	Yes	SALESTRANS_SALESREP. SALESTRANS_SALESREP_UUID	UUID calculated from PARTICIPANT
16	Sales Rep 3	String(20)	Yes	SALESTRANS_SALESREP. SALESTRANS_SALESREP_UUID	UUID calculated from PARTICIPANT
17	Sales Rep Type 1	String(20)	Yes	Not Directly Mapped	Used for UUID calculations for PARTICIPANT[1]
18	Sales Rep Type 2	String(20)	Yes	Not Directly Mapped	Used for UUID calculations for PARTICIPANT[1]
19	Sales Rep Type 3	String(20)	Yes	Not Directly Mapped	Used for UUID calculations for PARTICIPANT[1]
20	Rank 1	Number	Yes	SALESTRANS_SALESREP. RANK	
21	Rank 2	Number	Yes	SALESTRANS_SALESREP. RANK	
22	Rank 3	Number	Yes	SALESTRANS_SALESREP. RANK	
23	Split 1	Decimal	Yes	SALESTRANS_SALESREP. SPLIT_PERCENT	

Table 20. Transaction Header File Fields

No.	Field Name	Data Type and Size	NULL Allowed?	Mapped field in ICM	Comments
24	Split 2	Decimal	Yes	SALESTRANS_SALESREP. SPLIT_PERCENT	
25	Split 3	Decimal	Yes	SALESTRANS_SALESREP. SPLIT_PERCENT	

1. The constant values for the Sales Rep Type 1, Sales Rep Type 2, and Sales Rep Type 3 mappings are participant.customer, participant.channelPartner, and participant.employee. You provide those values in the input file. Those values are used in the mappings to look up the participant_uuid from the participant table using the participant code.

Transaction Line File

The transaction line file name is dataset1_line_flat.csv, as used in Informatica mappings. You can change the name as needed. It is a comma-separated text file with field names in first row. These fields are described in Table 21.

Table 21. Transaction Line File Fields

No.	Field Name	Data Type and Size	NULL Allowed?	Mapped field in ICM	Comments
1	Transaction Number	String(20)	No	SALESTRANS. SALESTRANS_NUMBER	Must be unique for *all* transaction headers for an Operating Unit. Connects header records to line records.
2	Line Number	Number	No	SALESTRANS_LINE. SALESTRANS_LINE_NUMBER	Must be unique for a Transaction Number for a Line.
3	Line Date	Date in MM/DD/YYYY format	No	SALESTRANS_LINE. LINE_DATE	
4	Line Type	String(30)	No	SALESTRANS_LINE. SALESTRANS_LINE_TYPE_UUID	Must exist in SALESTRANS_LINE_TYPE table. UUID is calculated with this field.
5	Line Description	String(512)	Yes	SALESTRANS_LINE. DESCRIPTION	
6	Product Code	String(30)	Yes	SALESTRANS_LINE.product_uuid	UUID calculated from product
7	Measure Code	String(30)	Yes	SALESTRANS_LINE.measure_uuid	UUID calculated from measure

Table 21. Transaction Line File Fields

No.	Field Name	Data Type and Size	NULL Allowed?	Mapped field in ICM	Comments
8	Sales Amount	Decimal	Yes	SALESTRANS_LINE_READING. ATTRIBUTE_VALUE_NUMBER	Profile Attribute must exist in ICM.
9	Quantity Sold	Decimal	Yes	SALESTRANS_LINE_READING. ATTRIBUTE_VALUE_NUMBER	Profile Attribute must exist in ICM.
10	Expected Ship Date	Date in MM/DD/YYYY format	Yes	SALESTRANS_LINE_READING. ATTRIBUTE_VALUE_NUMBER	Profile Attribute must exist in ICM.
11	Discount Code	String(20)	Yes	SALESTRANS_LINE_READING. ATTRIBUTE_VALUE_STRING	Profile Attribute must exist in ICM.
12	Discount Flag	String(20)	Yes	SALESTRANS_LINE_READING. ATTRIBUTE_VALUE_STRING	Profile Attribute must exist in ICM.
13	Sales Rep 1	String(20)	Yes	SALESTRANS_LINE_SALESREP. SALESTRANS_SALESREP_UUID	UUID calculated from PARTICIPANT
14	Sales Rep 2	String(20)	Yes	SALESTRANS_LINE_SALESREP. SALESTRANS_SALESREP_UUID	UUID calculated from PARTICIPANT
15	Sales Rep 3	String(20)	Yes	SALESTRANS_LINE_SALESREP. SALESTRANS_SALESREP_UUID	UUID calculated from PARTICIPANT
16	Sales Rep Type 1	String(20)	Yes	Not Directly Mapped	Used for UUID calculations for PARTICIPANT
17	Sales Rep Type 2	String(20)	Yes	Not Directly Mapped	Used for UUID calculations for PARTICIPANT

Table 21. Transaction Line File Fields

No.	Field Name	Data Type and Size	NULL Allowed?	Mapped field in ICM	Comments
18	Sales Rep Type 3	String(20)	Yes	Not Directly Mapped	Used for UUID calculations for PARTICIPANT
19	Rank 1	Number	Yes	SALESTRANS_LINE_SALESREP. RANK	
20	Rank 2	Number	Yes	SALESTRANS_LINE_SALESREP. RANK	
21	Rank 3	Number	Yes	SALESTRANS_LINE_SALESREP. RANK	
22	Split 1	Decimal	Yes	SALESTRANS_LINE_SALESREP. SPLIT_PERCENT	
23	Split 2	Decimal	Yes	SALESTRANS_LINE_SALESREP. SPLIT_PERCENT	
24	Split 3	Decimal	Yes	SALESTRANS_LINE_SALESREP. SPLIT_PERCENT	
25	Event Date	Date in MM/DD/YYYY format	No	SALESTRANS_EVENT. EVENT_DATE	
26	Event Type	String(30)	No	SALESTRANS_EVENT. SALESTRANS_EVENT_TYPE_UUID	Must exist in SALESTRANS_EVENT_TYPE. UUID is calculated with this field.

Informatica Objects for the ICM Transaction Data Model

This topic describes the Informatica objects for the Siebel ICM sales transaction data model.

Workflow, Session, and Mappings

Informatica objects used for the Sales Transaction Load process include workflows and their associated sessions, mappings, and database tables. Table 22 lists the sessions in the order (from top to bottom) in which the corresponding workflow runs them.

Table 22. Workflows, Sessions, Mappings, and Tables

WorkFlow[1]	Session	Mapping	Target Tables	Comments
wf_SALESTRANS	s_LOAD_SALESTRANS	m_LOAD_SALESTRANS	SALESTRANS	SALESTRANS is loaded first by this mapping.
	s_LOAD_SALESTRANS_ SALESREP	m_LOAD_SALESTRANS_ SALESREP	SALESTRANS_ SALESREP	This session requires a successful load of SALESTRANS.
	s_LOAD_SALESTRANS_READING	m_LOAD_SALESTRANS_ READING	SALESTRANS_ READING	This session requires a successful load of SALESTRANS.
wf_SALESTRANS_LINE	s_LOAD_SALESTRANS_LINE	m_LOAD_SALESTRANS_ LINE	SALESTRANS_ LINE	This session requires a successful load of SALESTRANS.
	s_LOAD_SALESTRANS_ LINE_ EVENT	m_LOAD_SALESTRANS_ LINE_EVENT	SALESTRANS_ EVENT	This session requires a successful load of SALESTRANS_LINE.
	s_LOAD_SALESTRANS_LINE_READING	m_LOAD_SALESTRANS_ LINE_READING	SALESTRANS_ LINE_READING	This session requires a successful load of SALESTRANS_LINE.
	s_LOAD_SALESTRANS_LINE_REP	m_LOAD_SALESTRANS_ LINE_REP	SALESTRANS_ LINE_REP	This session requires successful load of SALESTRANS_LINE.

1. The workflows should be run in order shown in the table; that is, first run the wf_SALESTRANS workflow, then run the wf_SALESTRANS_LINE workflow.

Mapping Parameters

All the mappings listed in Table 22 have one mapping parameter named $$OPERATING_UNIT_CODE. This parameter stores the Operating Unit name. This Operating Unit name must match the name of the Operating Unit you want to populate with data. You can change the Operating Unit name as necessary.

External Procedure

The system uses an external procedure object named INF_GenerateUUID.dll to generate UUID (Universally Unique Identifier) values. It is recommended that you place this object in the Informatica folder $PMExtProcDir. The External Procedure Transformation in the header record's Transaction Type field that uses this procedure object offers different UUID options for Windows and UNIX systems, as follows:

- **Windows.** The External Procedure Transformation takes as its input one of two strings, either Short or Long. This string is not case-sensitive. Depending on the input, the transformation generates either a short UUID or a long (36 character) UUID.
- **UNIX.** The External Procedure Transformation takes as its input only one string, Short. This string is not case-sensitive. The transformation generates a short UUID.

NOTE: Short UUIDs are of variable length. The minimum length is 10 characters. They can be 11 or 12 characters, depending on how many are created at the same system time.

For more information about UUIDs, see "Universally Unique Identifiers" on page 224.

ICM Sales Transaction Data Model

Sales Transactions and their related data are stored in a set of tables. These tables are connected by foreign key relationships, and they depend on data in other tables. The tables are listed and described in Table 23.

Table 23. ICM Sales Transaction Tables

Table	Comments
SALESTRANS	Stores the base sales transaction header data, including the transaction number as defined by the external system.
SALESTRANS_SALESREP	Represents sales representatives associated with the sales transaction header. Each header sales representative has one record in this table. For example, if a header has three sales representatives, this table will contain three records corresponding to the single record in SALESTRANS.
SALESTRANS_READING	Represents the readings, or profile attributes, associated with the sales transaction header. Each header profile attribute has one record in this table. For example, if a header has three profile attributes, this table will contain three records corresponding to the single record in SALESTRANS.

Table 23. ICM Sales Transaction Tables

Table	Comments
SALESTRANS_LINE	Stores one line of a sales transaction. Has a foreign key relationship to SALESTRANS. Every record in SALESTRANS_LINE must correspond to a record in SALESTRANS.
SALESTRANS_LINE_READING	Represents the readings, or profile attributes, associated with one line. Each line profile attribute has one record in this table. This table has profile reading data at the Transaction Line level.
SALESTRANS_LINE_SALESREP	Represents the sales representatives associated with one line. Each sales representative associated with the line has one record in this table. This table has sales representative data at the Transaction Line level.
SALESTRANS_EVENT	Represents a sales transaction event. This table has event data at the Transaction Line level. It has an event type and an event date. Population of this table is critical for the sales crediting service and for all subsequent services.

Transaction Loader Limitations

This topic describes some limitations to the types of data that the Transaction Loader can process.

Sales Transaction Adjustments

This Transaction Loader does not support adjustments or updates to existing transactions.

Siebel ICM has an adjustment model for tracking changes to sales transactions. Adjustments are defined as updating fields of a sales transaction at the header, line, or event level. This includes, but is not limited to, updating profile attribute values and canceling events.

To retain an audit trail of modifications to the sales transaction, the system tracks adjustments to a sales transaction in a separate table. This tracking logic is implemented in the business logic layer of the application.

Retroactive Processing

This Transaction Loader does not support Retroactive Processing of transactions. Do *not* try to add or change transactions retroactively with the Transaction Loader.

Retroactive processing is triggered by creating or adjusting sales transaction events in a closed period, among other things. The reference implementation inserts sales transaction events only into an open period.

Best Practice for Informatica Analytics Updates

You can configure Informatica workflow mappings to improve the performance of the Update Analytics process for the initial data transfer. It is recommended that you change the following workflows, which process the highest volumes of data:

wf_attribute_def
wf_f_credit
wf_f_credit_reading
wf_f_earning
wf_f_earning_reading
wf_f_salestrans_event
wf_f_salestrans_line
wf_f_salestrans_line_reading
wf_f_salestrans_line_salesrep

For information about the Update Analytics process, see "Update Analytics Service" on page 138.

Configuring Informatica Workflows to Improve Performance

To configure the Informatica workflows to improve Update Analytics data transfer performance for the initial data transfer *only*, follow this procedure for each workflow.

To configure Informatica workflows to improve performance

1. In Informatica, open a workflow and note the session name the workflow is using.
2. Edit the session by doing one of the following:
   - If the session is reusable, find it in your Sessions folder and open it for editing.
   - If the session is not reusable, right click it and choose Edit from the context menu.
3. Select the Properties tag.
4. In the General Option section, change the value for Treat Source Rows As to Insert.
5. After the initial data transfer has been completed, repeat Step 1 through Step 3 and, in the General Option section, set the value for Treat Source Rows As back to Update.

CAUTION: If you do not change Treat Source Rows As from Insert back to Update after the initial loading, the updated data may not be transferred because of constraint violation errors in later UAS runs.

D Profiling Siebel Incentive Compensation Management with JProfiler

This appendix describes the settings you would use in JProfiler for ICM compatibility. It includes the following topics:

- "About JProfiler and ICM" on page 235
- "Specifying JProfiler Settings for ICM" on page 235
- "Guidelines for Profiling ICM with JProfiler" on page 236

About JProfiler and ICM

JProfiler is a Java memory and CPU profiling application that allows you to view the JVM and discover what areas of your application are consuming the most CPU time and memory resources. You can download an evaluation copy of JProfiler from the following Web page:

`http://www.ej-technologies.com/download/jprofiler/trial.php`

This appendix specifies the JProfiler settings likely to give you the most consistent and reliable profiling results for Siebel ICM. These settings provide a starting point from which you can profile your ICM instance. They may require adjustment for your specific ICM deployment. This appendix also suggests some guidelines for profiling ICM with JProfiler when it is set up.

Specifying JProfiler Settings for ICM

This topic specifies field values and option selections that make JVM profiling compatible with ICM. For detailed instructions on setting up and running JProfiler, and for reference information about values and options, consult the JProfiler documentation.

To specify JProfiler settings for ICM

1. Launch JProfiler.
2. From the Quick Start menu, do the following:
   a. Choose An application server, locally or remotely.
   b. Select your application server.

      Valid options for ICM are JBoss 3.x and IBM WebSphere 5.
   c. Select On this computer.
   d. Identify a location, as follows:
      - If your application server is JBoss, select run.bat.
      - If your application server is WebSphere, select server.xml.

3 In the Application Settings screen, complete the fields and selection options.

4 In the Profiling Settings screens, complete the fields and selection options.

Selections and values for the items that affect ICM profiling are described in the following table.

Item	Comments
Call tree collection method	Dynamic Instrumentation **CAUTION:** The Sampling with Frequency and Full Instrumentation settings are *not* recommended. Sampling with Frequency takes incremental snapshots of the heap. As a result, it can detect only the net heap changes between samples. This setting misses the fine detail required to profile ICM, and it sometimes generates inaccurate results. Use the Sampling with Frequency setting only when nothing else works, and then use the results only as a high-level picture of JVM activity. Full Instrumentation allows you to profile inside the jdk classes, but it is extremely costly in terms of performance time.
Active filter sets	Inclusive Filters
Only resolve calls from	com.motiva
Disabled profiling features	Disable monitor contention views **NOTE:** Because the monitor contention views are disabled, some of the Thread View and VM Telemetry views will not be displayed. This should not be a problem for most deployments.
Allocation monitor	Show allocations resolved for each class
Console settings	Native console (Windows only)
VM life cycle control	Keep VM alive
Call stack trees and thread history	10s
Tables and graphs	5s

Guidelines for Profiling ICM with JProfiler

ICM is a large application that consumes all available JVM resources during service runs. Additionally, running the profiling program itself adds a significant amount of overhead. Therefore, to profile ICM, you must specify profiling and application parameters that allow both applications to run in the available memory space.

It is common for the application JVM to freeze or exit during profiling. The frequency with which the application JVM freezes is directly related to the amount of data you try to profile. After the JVM exits, JProfiler cannot collect memory statistics, which invalidates the profiling session. To minimize freezing or exiting of the JVM, follow these guidelines:

- Do not try to profile CPU and memory in the same profiling session.

  Even if you are seeing out-of-memory errors, do one run with CPU profiling and then repeat the run with Memory Profiling turned on.

- Turn off all the metrics you do not need.

  The more metrics you collect, the greater the chance that the JVM will freeze or exit prematurely. For example, if you are trying to diagnose out-of-memory exceptions, you do not need information on thread contention. In this case, disable the monitor contention views setting.

- On a machine with 2GB of RAM, a JVM -Xmx setting of -Xmx768m is recommended as offering the best combination of profiling reliability and production verity.

  CAUTION: Increasing the JVM -Xmx setting to its maximum value (2000m on 32 bit Windows) does not necessarily improve out-of-memory errors diagnoses. Setting this value too high increases the chance that the application JVM will exit when it encounters out-of-memory errors. Setting the value too low does not adequately represent the application's production environment, and may cause out-of-memory errors that the application might not encounter in production. (However, this can be useful to reproduce out-of-memory errors more quickly and at lower RAM consumption levels.)

- Do not try to use the “Heap snap shot” feature.

  Using this feature will crash the ICM VM. Siebel ICM uses too much memory for this feature to work properly.

E Open Integration Framework Reference

This appendix describes the interface to ICM-importable data entities. It is meant to help implementors and integrators design import and export files for ICM. This appendix includes the following topics:

- "About ICM's Open Integration Framework" on page 239
- "Import Objects Schemas" on page 242

About ICM's Open Integration Framework

The ICM Open Integration Framework accepts XML documents whose structures are defined by XML schemas. The following topics provide general description information about those schemas:

- "Basic XML Schematics" on page 239
- "Profile Attributes and Custom Attributes in XML Files" on page 240
- "Dependency-Sorted Import Services" on page 242
- "Date and Time Formats" on page 242
- "About Translating XML Files Into ICM Format" on page 242

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. Between this tag and the </recordset> tag are all the files that will be imported.

The <*TYPE*> tag indicates the beginning of a record of a specific type. In a file, *TYPE* is replaced by a record type such as *EMPLOYEE* or *CUSTOMER*. The paired </*TYPE*> tag indicates the end of the record. Between these two tags are any number of <*field*> lines, indicating specific data fields for each record. After the </*TYPE*> tag, another <*TYPE*> tag begins the next record. 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 in a record, such as *code* or *lastName*. *value* may be any valid value for that data field; for example, *lastName* accepts any text string for an employee's last name. The number of fields for a record may vary according to the type of import file, whether the field is required, and so on.

Some types of XML files may contain subrecords. Such a file might look like this:

```
<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>
```

The <*SUBTYPE*> tag represents any kind of data within a record that might require its own subset 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 careers, and ICM 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.

NOTE: The XML code created in OU import and export files is not the same format as in other services. <motiva></motiva> tags (instead of <recordset></recordset> tags) are used in the OU Import and Export functionality only. For information about exporting and importing operating unit data sets, see the chapter on exporting and importing operating units in *Siebel Incentive Compensation Management Administration Guide*.

Profile Attributes and Custom Attributes in XML Files

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

```
<customField readingType="Name" typeofvalue="Value1"/>
```

NOTE: Profile attributes in XML code are also called profile attributes in ICM's user interface. For more information, see "Profile Attributes" on page 49.

The readingType tag indicates the name of the profile attribute. Within the quotes you must specify the name of the profile attribute. The next attribute indicates the type of value contained in the reading type. The tag *typeof*value is replaced by the type of data stored in the profile attribute, such as stringvalue or numericvalue. The actual value is specified within the quotes.

The tag for a custom attribute follows this pattern:

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

The name tag indicates the name of the custom attribute, and the value tag indicates the value stored in that field.

NOTE: In both cases, there is no closing tag for either tag. The slash mark (/) at the end of each profile attribute or custom attribute line indicates the end of that tag.

Any number of profile attributes or custom attributes may be contained within a record. You can add them manually to an XML file. To transform a file exported from an external system into the proper format for import into ICM, configure the export function of the external systems to send the appropriate data into profile attributes or custom attributes, or use an external XML adapter.

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

The Channel Partner, Customer, Product, Organization, and Territory import services can sort an imported XML file based on parent-child relationships. This causes the XML entities to be imported in the correct top-down order. For example, parents are imported and created in ICM before their children. The sort takes place during the import process. No change to the order or format of the XML files is necessary.

Date and Time Formats

This topic describes date and time formats in ICM XML schematics.

The XML Schema supports both date and time data types. Date and time formats are fixed on import. After the data is in the system, however, ICM can display and manipulate that data in whatever locale the user has selected.

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

```
CCYY-MM-DD
```

Where CCYY is the four-digit year, MM is the month number (with leading zero as needed), and DD is the day number (with leading zero as needed). For example, 2001-05-03 equals May 3, 2001. This format is required by the underlying data type xsd:date defined in the XML Schema. If a date in the import file is not in this format, the system will reject its record.

For the official description of this format, see http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#date.

About Translating XML Files Into ICM Format

For instructions on translating XML files into ICM XML format, see the README.txt file in the <CDROOT>/FileConvertor directory on your Siebel ICM installation CD.

Import Objects Schemas

The structure of ICM-importable XML files is defined by XML schemas. An XML schema exists for each type of object listed in this section. This section provides specific descriptions of those objects' schemas.

This section includes the following topics:

- "Employees, Job History, and Jobs" on page 243
- "Sales Transactions" on page 246
- "Credits" on page 250
- "Goals" on page 252
- "Products and Product Hierarchy" on page 253

- "Customers" on page 255
- "Organizations and the Organization Hierarchy" on page 259
- "Territories and Territory Hierarchy" on page 260
- "Channel Partners" on page 265
- "Exchange Tables and Rates" on page 268
- "Matrix Expressions" on page 268

NOTE: In the topics of this section, required fields are in **bold** text. Cardinality, or number of occurrences of the field, is in braces: {one or more}. No braces means that the field can occur only once. Attributes of an element are included in pointed brackets: <>.

Employees, Job History, and Jobs

The Employee and Job objects implement Extended Attributes.

Profile Attributes 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.

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 list includes 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 include SupervisorIDs, which are Employee codes, the supervisor you are referencing must already be present in the Employee table. There is a dependency, so you may need to import in two passes.

- Role (only if importing user account information)

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

Employee Schema

The employee schema is defined in employee.xsd. The following elements (or fields) are defined in the XML Schema:

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
  - Code - (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 Employee Schema XML Document

```
<?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@motiva.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

Sales Transactions are also known as Invoices. 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 Profile Attributes 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 ICM, the Sales Transaction Item, Line, and Event objects replace INVOICM and INVOIC. Line Types and Event Types are not defined by the application. You can define Line Types and Event Types according to the requirements of your organization. For example, if the date on which a contract is signed is significant for crediting purposes, you can define an event type called Contract Signed.

Any number of sales reps may be associated at both the master (or item) level and the 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. For more information, see "Sales Transaction XML Schema" on page 248. The adjustType values are checked and handled first at the item level, then for each line in the item, and finally for each event in the line.

The rules for specifying adjustType are as follows:

Item Level

If adjustType is not present, assume this is a new sales transaction and try to create it, along with all lines, events, readings, and sales reps. Do not check adjustType at the line or event level. This is the 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 try to adjust its associated readings or other header and item information, then check whether 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 check the adjustType at the line level. No adjustment will be performed at the item level, including readings and sales reps associated with the item. Use adjust_item to adjust at the item level.

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 check the line's events for adjustments.

NOTE: If you create a new transaction, it is not necessary to specify adjustType at the line or event level.

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 only the changes. Instead, enter the entire new set of sales rep information.

In most cases, a header-level adjustType is sufficient, because you will reimport the entire sales transaction with an adjustment. Only rarely will you provide deltas to an existing sales transaction.

NOTE: The data included in an adjusting XML transaction will replace existing data for the entity being replaced. 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 Sales Transaction Servicer creates a new copy of the header or line and cancels the original one. This can result in a 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 uses the header profile and line profile to verify that the correct header Profile Attributes and line Profile Attributes were included in the import. The line type, however, is not required in the transaction XML because it is an attribute of the line.

Dependent objects, which must be present in the database before importing Sales Transaction data, are as follows:

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

Sales Transaction XML Schema

The Sales Transaction schema is defined in salesTrans.xsd. The following elements (or fields) are defined in the XML Schema:

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 Sales Transaction Schema XML Document

```
<?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

Credits are 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 profile attributes.

Core Entities

- Credit
- Credit Reading

Dependent Entities

- Measure (required)
- Profile (if importing readings)
- Profile Attribute (if importing readings)
- Participant (Employee, Customer, 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 Profile Attributes. Before importing Credits, you must design the different Measures and their Profiles and Profile Attributes, and create them in the UI.

A credit may be assigned to 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 Schema

The Credit schema is defined in 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 (also known as readings) must match the Profile Attributes 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 by putting quotes around it. Optionally, you can indicate a currency.

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 set up the desired Profile Attributes, Profiles, and Measures.

Core Entites

- Goal
- Goal Reading

Dependent Entities

- Measure (required)
- Profile
- Profile Attribute
- 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 Profile Attributes. Before importing Goals, you must decide what the Measures and their Profiles and Profile Attributes will be, and create them in the UI.

A Goal may be assigned to 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 Schema

The Goal schema is defined in 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 (also known as readings) must match the Profile Attributes 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 by putting quotes around it. Optionally, you can indicate a currency.

Sample Goal Schema 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>
```

Products 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).

You cannot import a product hierarchy. You must import the individual products, and then set up the hierarchy in ICM.

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 Schema

The Product schema is defined in 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 Product Schema 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>
```

Customers

Customers are also known as Accounts.

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, however, may have CustomerLocations with the same codes.

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.

Circular Dependencies

If you are importing a customer with customer location records, the customer.shipToLocationID and customer.soldToLocationID fields must be null. To set these fields, you must create a second import file containing the customer code and the shipToLocationID and soldToLocationID values. The import service will update the references.

You import this information in two passes because the Customer object must be created first so that 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 you create the Customer object. You can update the Customer object in a second pass with shipTo and soldTo information.

NOTE: You do not have to populate these two fields *unless* your crediting rules depend on the existence of this information.

CustomerLocation Codes

For any given customer, no two customer locations can have the same code. This applies to CustomerContact as well. You cannot include the same customer location code twice in an import record for a given customer. This will cause the Import Service to fail.

Customer Schema

The Customer schema is defined in 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 Schema 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>
```

Organizations and the Organization Hierarchy

The Organization object is extensible, but you cannot import extensible attributes or an Organization Hierarchy. You can, however, import the organization's level in the hierarchy.

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 because it does not yet exist. If you import employees first, you cannot include the organization ID in the employee Job structure because it does not yet exist.

To address this issue, it is recommended that you import organizations before employees. The Organization Import service creates a stub record for the manager's Employee ID. Then, import the full Employee records with the Employee Import service.

Dependent Objects

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

Organization Schema

The Organization schema is defined in 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 Organization Schema 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>
```

Territories and Territory Hierarchy

Territory is one of the more complex import entities, if you use the qualifier mechanism. You can define extended attributes on the Territory object.

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 the time of crediting to determine which territories, or which participants assigned to territories, should receive credit for a transaction. ICM defines a set of qualifier attributes that you can use to create expressions. These attributes are as follows:

- product
- customer
- city

- county
- state
- province
- zip
- country
- areaCode

For example, you can define a territory qualifier 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.

You can build Territory Qualifier Conditions using transaction line or header readings (also called profile attributes), allowing expressions like the following:

```
salesAmount >= 10,000
```

where salesAmount is a profile attribute defined in the transaction line or header profile.

Updating Territories

The Territory Import Service does not perform incremental updates. That is, it cannot compare the contents of the import record with the existing qualifiers, conditions, and assignments to see what changed. It only 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 completely replaces 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
  - Allows more than one participant can be assigned to a territory
  - Allows Participants to have Roles in a territory
- Territory Qualifier (a container for Territory Qualifier Conditions)
  - Territory Qualifier Condition

Dependent Entities

Before you can import territories, you must create these entities.

- Territory Role

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

Territory Schema

The Territory schema is defined in 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>

The <condition> element that 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 indicates 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 in the preceding paragraphs, like zip, product, or customer. You can omit salesTransContext and readingTypeCode from the XML document for this condition.

If the <condition> type is reading, the condition is based on the readingTypeCode and salesTransContext. Both these elements must be included in the XML. The import service creates a new Territory Qualifier Attribute if it does not 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 you use the like and notLike operators, <value1> may include a Perl5-style regular expression. For more information on Perl 5 regular expressions, go to www.perl.com or see the book *Programming Perl*.

If you use the inList or notInList operator, make <value1> a string of comma-separated values, like this:

```
a, b, c, z
```

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

Sample Territory Schema 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 creates a Territory definition with a condition based on the ZIP Code. The second <Territory> record creates conditions based on ZIP Code and salesAmount, a reading type that is expected on the header profile at the time of territory qualification evaluation.

Channel Partners

The Channel Partner entity is extensible.

Channel Partners are participants. Make sure to specify <participantFlag>true</participantFlag> so that the import service creates 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

The Channel Partner schema is defined in 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 Schema 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 Tables and Rates

The exchange table and exchange rate are not extensible. They do not support profiles and readings.

You must specify exchange rates in both directions, because the import does not perform rate calculations. For example, if you try to import USD->HKD = "1.5", the system does not add HKD->USD=.66.

Dependent Objects

- Currency

Exchange Table Schema

The Exchange Table schema is defined in 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 Exchange Table Schema 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 Expressions

Matrix Expressions are not extensible. They do not support readings.

For 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

An example of a matrix expression is shown in Table 24.

Table 24. Example XML Matrix Expression

Origin Cell MyRowLabel	MyColumnLabel: A	MyColumnLabel: B	MyColumnLabel: C
0-10000	1	2	3
10000-15000	4	5	6
15000-999999	7	8	9

Sample Matrix Variation Attributes

- 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, also called dimensions, for specific job, organization, 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 for column and row entries, except for the import service. You can include rounding rule codes in the import XML, but the import service will ignore them.

Matrix Expression 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

The following general validations are applied to all cases of creating or updating a matrix.

- If expression variation = Default, org/job/participant codes in the import record will be ignored.
- If expression variation = Org/Job/Participant, the matrix variation will be rejected.
- The given organization, job, and participant code exists for the operating unit.
- Missing cell values will be set to 0.0.
- col and row rounding rule codes must correspond to rounding rules valid in the OU.
- The (0,0) cell will be created automatically. Any data passed in for this cell will be ignored.
- Date or date range row and column headers should be in the form MM/DD/YYYY.
- Date values will be parsed into Date objects and stored as Date.getTime(). Invalid dates will cause the entire matrix variation to be rejected.
- If matrix type (apply to type) is rows, the colCount, colHeaderType, colLabel, colRoundingRuleID elements will be ignored.
- 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 Matrix Expression XML Document

```
<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>
```

Guidelines for Creating Matrix Expression XML Documents

When creating matrix expression XML documents, observe the following guidelines:

- Use the headerFromValue and headerToValue attributes of the <cell> element to specify row and column header values.
- Use headerToValue to specify the upper boundary 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. Although resultValue will be quoted because it is an attribute, the value will be parsed to a number (a double primitive).
- The XML schema contains enumerations for all appropriate attributes and elements. Validate your XML document against the schema to make sure it will be imported correctly.
- You must always include a matrix variationType="default" when creating a new matrix. If you import new variations for an existing matrix, you do not need to include a default.
- For a cell representing a column or row header, use the headerFromValue attribute. Use headerToValue only if the row header type or column header type is of range type number_range, string_range, or date_range.

F XML Service Batch Schema

The following is the XML schema definition that describes Oracle's Siebel Incentive Compensation Management (ICM) product Service Batch xml.

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by Kjirsten Koka (Siebel Systems, Inc)
-->
   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
   attributeFormDefault="unqualified">
      <xs:element name="motiva">
         <!--
                     A Siebel ICM service batch definition.  The 'motiva' root identifies this document
as an xml
         file that the Siebel ICM can process.
      -->

   <xs:complexType>
      <xs:sequence>
         <xs:element name="serviceBatch" maxOccurs="unbounded">
                        <!-- Declare a batch of services to run -->
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="runAsUsername" type="xs:string">
                                 <!-- The user name to run all services as -->
                              </xs:element>
                  <xs:element name="runAsPassword" type="xs:string">
                                 <!-- The password for the runAsUsername -->
                              </xs:element>
                  <xs:element name="beginPeriod">
                                 <!-- The first period this batch processes -->
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element ref="calendarYear" minOccurs="0">
                                                      <!-- The calendar year code -->
                                          </xs:element
                        </xs:sequence>
                                          <!-- NOTE: use either periodNumber or
absolutePeriodNumber, not both -->
                                             <!-- the relative period number in the calendar year -
->
                        <xs:attribute name="periodNumber" type="xs:integer" use="optional"/>
                                          <!-- the absolute period number in the calendar -->
                        <xs:attribute name="absolutePeriodNumber" type="xs:integer" use="optional"/>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="endPeriod">
                                 <!-- For batch display purposes: the last period this batch
processes -->
                     <xs:complexType>
                        <xs:sequence>
                                          <!-- The calendar year code -->
                           <xs:element ref="calendarYear" minOccurs="0"/>
                        </xs:sequence>
                                          <!-- NOTE: use either periodNumber or absolutePeriodNumber,
not both -->
                                          <!-- the relative period number in the calendar year -->
                        <xs:attribute name="periodNumber" type="xs:integer" use="optional"/>
                                             <!-- the absolute period number in the calendar -->
                        <xs:attribute name="absolutePeriodNumber" type="xs:integer" use="optional"/>
                     </xs:complexType>
                  </xs:element>
                                    <!-- Should the batch stop if any service error occurs? -->
```

```
                  <xs:element name="haltOnServiceError" type="xs:boolean" minOccurs="0"/>
                                    <!-- Comma delimited list of email addresses to notify of batch
events: start, stop, error -->
                                    <!-- NOTE: The notification feature is not implemented with
Siebel 7.7 -->
               <xs:element name="notifyEmailAddresses" type="xs:string" minOccurs="0"/>
                  <!-- default log level.  Optional. Valid values for type 'LogLevel' are defined below.
-->
                       <xs:element name="defaultLogLevel" type="LogLevel" minOccurs="0"/>
                           <!-- Identifies this batch as generated by the system to process a retro
revision with this code.
                                For display purposes only -->
               <xs:element name="retroRevision" minOccurs="0">
                  <xs:complexType>
                     <xs:attribute name="code" type="xs:string" use="required"/>
                  </xs:complexType>
               </xs:element>
                                <!-- Define a service to run as part of this batch -->
               <xs:element name="serviceBatchItem" maxOccurs="unbounded">
                  <xs:complexType>
                     <xs:sequence>
                                          <!-- Define the service code, calendar year, period,
input, and optimization level for this service -->
                        <xs:element name="serviceDefinition">
                           <xs:complexType>
                              <xs:attribute name="code" type="ServiceCode" use="required"/>
                           </xs:complexType>
                        </xs:element>
                                       <!-- Define the period in which this service should be
executed.  Overrides the period defined above. -->
                        <xs:element name="period">
                           <xs:complexType>
                              <xs:sequence>
                                 <xs:element ref="calendarYear" minOccurs="0"/>
                              </xs:sequence>
                              <xs:attribute name="periodNumber" type="xs:integer" use="optional"/>
                              <xs:attribute name="absolutePeriodNumber" type="xs:integer"
use="optional"/>
                           </xs:complexType>
                        </xs:element>
                                          <!-- An integer indicating the order of this service
within the batch.
                                                 Created when the system exports this file.  By
default the system executes the services in
                                                 the order they are defined. -->
                        <xs:element name="rank" type="xs:integer" minOccurs="0"/>
                                          <!-- If a service supports optimized processing, specifies
the level of optimization the service should use. -->
                                       <!-- NOTE: not supported for 7.7 -->
                        <xs:element name="optimizationLevel" type="xs:string" minOccurs="0"/>
                                       <!-- Overrides the above defaultLogLevel setting -->
                        <xs:element name="logLevel" type="LogLevel" minOccurs="0"/>
                                       <!-- Specify service-specific input here -->
                        <xs:element name="input" type="xs:string" minOccurs="0"/>
                                       <!-- For testing purposes only.  You can specify a JUnit
test suite to execute in response to
                                                            this service event -->
                        <xs:element name="event" minOccurs="0" maxOccurs="unbounded">
                           <xs:complexType>
                              <xs:sequence>
                                 <xs:element name="testSuite" minOccurs="0" maxOccurs="unbounded">
                                    <xs:complexType>
                                                               <!-- The fully qualified class
name of the JUnit test suite you want to execute.  All
                                                                     testXXX methods will be
executed.  For example:
    'com.motiva.ce.service.test.rollup.PostFinishTest' -->
                                       <xs:attribute name="suiteName" type="xs:string" use="required"/>
                                                                <!-- If the JUnit test suite
defined in suiteName requires an external answer file,
```

```
                                                                                specify the absolute or
relative path of the file here -->
                                    <xs:attribute name="dataFile" type="xs:string" use="optional"/>
                                 </xs:complexType>
                              </xs:element>
                           </xs:sequence>
                                                              <!-- The name of the system event to capture.  Valid
system event names are defined below. -->
                                       <xs:attribute name="name" type="SystemEvents" use="required"/>
                                    </xs:complexType>
                                 </xs:element>
                              </xs:sequence>
                           </xs:complexType>
                        </xs:element>
                     </xs:sequence>
                              <!-- The unique code used to describe this service batch -->
                        <xs:attribute name="code" type="xs:string" use="required"/>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
   <xs:simpleType name="LogLevel">
         <!-- valid values are: -->
            xs:restriction base='xs:string'>
                  <xs:enumeration value='DEBUG'/>
                  <xs:enumeration value='INFO'/>
                  <xs:enumeration value='WARN'/>
                  <xs:enumeration value='ERROR'/>
                  <xs:enumeration value='FATAL'/>
         </xs:restriction>
</xs:simpleType>
   <xs:simpleType name="ServiceCode">
                  <!-- The valid service code values are as follows: -->
                  <xs:restriction base='xs:string'>
                        <xs:enumeration value='ChannelPartnerImport'/>
                        <xs:enumeration value='CostCenterImport'/>
                        <xs:enumeration value='CreditImport'/>
                        <xs:enumeration value='CumulateService'/>
                        <xs:enumeration value='CustomerImport'/>
                        <xs:enumeration value='EarningCalculationService'/>
                        <xs:enumeration value='EarningSummarizationService'/>
                        <xs:enumeration value='EmployeeImport'/>
                        <xs:enumeration value='ExchangeRateImport'/>
                        <xs:enumeration value='FinalizePaymentService'/>
                        <xs:enumeration value='FormulaImport'/>
                        <xs:enumeration value='GoalExport'/>
                        <xs:enumeration value='GoalImport'/>
                        <xs:enumeration value='IncrementRetroRevision'/>
                        <xs:enumeration value='JobImport'/>
                        <xs:enumeration value='LocationImport'/>
                        <xs:enumeration value='MatrixCalcImport'/>
                        <xs:enumeration value='OperatingUnitExport'/>
                        <xs:enumeration value='OperatingUnitImport'/>
                        <xs:enumeration value='OrganizationImport'/>
                        <xs:enumeration value='PaymentExport'/>
                        <xs:enumeration value='PaymentImport'/>
                        <xs:enumeration value='PeriodCloseService'/>
                        <xs:enumeration value='PlanEligibilityService'/>
                        <xs:enumeration value='PlanEntityImport'/>
                        <xs:enumeration value='PlanImport'/>
                        <xs:enumeration value='ProductImport'/>
                        <xs:enumeration value='PurgePeriodDataService'/>
                        <xs:enumeration value='ResetScenarioService'/>
                        <xs:enumeration value='RetroBatchService'/>
                        <xs:enumeration value='RetroCumulateService'/>
                        <xs:enumeration value='RetroEarningCalculationService'/>
                        <xs:enumeration value='RetroEarningSummService'/>
                        <xs:enumeration value='RetroPeriodCloseService'/>
                        <xs:enumeration value='RetroPlanEligibilityService'/>
```

```
                    <xs:enumeration value='RetroRollupService'/>
                    <xs:enumeration value='RetroSalesCreditingService'/>
                    <xs:enumeration value='RollupService'/>
                    <xs:enumeration value='RuleSetImport'/>
                    <xs:enumeration value='SalaryGradeImport'/>
                    <xs:enumeration value='SalesCreditingService'/>
                    <xs:enumeration value='SalesTransactionImport'/>
                    <xs:enumeration value='SetupEntityImport'/>
                    <xs:enumeration value='SiebelCRMExtractService'/>
                    <xs:enumeration value='StepCalcImport'/>
                    <xs:enumeration value='TerritoryImport'/>
                    <xs:enumeration value='ThresholdCalcImport'/>
                    <xs:enumeration value='TransactionExport'/>
                    <xs:enumeration value='TransactionPurgeService'/>
                    <xs:enumeration value='TrialPaymentCalculationService'/>
                    <xs:enumeration value='UpdateAnalyticsService'/>
            </xs:restriction>
</xs:simpleType>
   <xs:simpleType name="SystemEvents">
     <!-- valid system events values are: -->
     <xs:restriction base='xs:string'>
            <xs:enumeration value='pre_extract'/>
            <xs:enumeration value='post_extract'/>
            <xs:enumeration value='pre_cleanup'/>
            <xs:enumeration value='post_cleanup'/>
            <xs:enumeration value='pre_retryCleanup'/>
            <xs:enumeration value='post_retryCleanup'/>
            <xs:enumeration value='pre_initprocessingcontext'/>
            <xs:enumeration value='post_initprocessingcontext'/>
            <xs:enumeration value='pre_prepare'/>
            <xs:enumeration value='post_prepare'/>
            <xs:enumeration value='pre_finish'/>
            <xs:enumeration value='post_finish'/>
     </xs:restriction>
</xs:simpleType>
   <xs:element name="calendarYear">
      <xs:complexType>
         <xs:attribute name="code" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Index

A

Absolute period 31
account
See customer object
Account Entity field mappings 159
actuals
See Credit Object
Actuate server
adding an ICM folder 178
granting access 178
adding
calendar year 33
child organizations 90
child products 95
content to dashboard page 183
dashboard pages 182
employee to a territory 116
ICM screen access to responsibilities 173
ICM screen tab 173
roles 39
Adjust privilege 37
advanced rolling file appender 203
aggregate operators 50
Application Log
about 199
advanced rolling file appender 203
appender definitions 202
appenders and categories 202
application rolling file appender 203
application server rolling file appender 204
Audit log appender 204
Boot log 200
category definitions 205
Console appender 203
LocalHost Access log 200
application rolling file appender 203
application server rolling file appender 204
Assign privilege 37
attributes
aggregate operators 50
extended attribute display types 62
extended attribute, about 61
extended attribute, creating 62
extended attribute, editing 64
inherent attributes 62
profile attribute data types 49
profile attributes, about 49
profile attributes, defining 52
Returnable fields 51
Reversible fields 51
setup overview 19
smart attributes, about 69
smart attributes, defining 70
Audit log appender 204

B

Boolean data 49
Boot log 200

C

calendars
about 31
calendar year, adding 33
custom and standard calendars, combining 32
custom calendar years, setting up 35
periods 31
Relative and Absolute periods 31
segment types 32
standard calendar years, setting up 33
Channel Partner XML schema
channelPartner.xsd 265
recordset 265
sample XML document 267
channel partners
about 97, 265
adding certifications 105
adding contacts 105
adding details 104
adding to a region 114
adding to a territory 116
channel partner schema 265
creating 103
dependent entities 265
importing 39
sample channel partner XML document 267
setting up process 103
channel segments
about 79
setting up 84
child
organizations, adding 90
products, adding 95

close period service 138
command line utility, File Adaptor 242
condition templates
about 73
customized 73
examples 211
Hdr Field (String) <operator> <value> 215
Line Field (String) <operator> <value> 213
Line Product Is Specified Product 211
setting up 74
configuration file, Publish Service 130
configuration properties, setting 153
configuring ICM
access, about 23
access, setting up 24
ongoing processes 21
roadmap 20
Console appender 203
context 20
copying ICM Web templates to CRM 168
cost centers
about 79
setting up 83
CPU profiling
See profiling
Create privilege 37
creating
dashboard 181
extended attributes 62
ICM screen tab objects 168
service batch 140
Credit object
about 250
core entities 250
credit XML schema 251
dependent entities 250
credit recipient templates
Line Rep of Given Rank 216
Line Rep of Given Rank's Organization at Given Level 218
credit XML schema
credit.xsd 251
recordset 251
sample Credit XML document 251
CRM-to-ICM integration
See Siebel CRM-to-ICM integration
CRM-to-ICM single sign-on
See single sign-on
Cumulate Service
about 137
currencies
about 43
conversion options, changing 46
conversions 43
currency codes, creating 44
exchange rates, importing 45
exchange rates, setting 45
setting up process 43
unit currencies, assigning 44
Currency data, about 49
custom calendar years
custom and standard calendars, combining 32
setting up 35
custom segment types 32
Customer Code smart attribute 209
Customer object
core entities and dependent entities 255
customer XML schema 255
sample customer XML document 258
customer XML schema
customer.xsd 255
recordset 255
sample customer XML document 258
customers
about 98
adding contacts 109
adding customers 106
adding details 107
adding locations 108
importing 39
setting up process 106
customized condition templates 73

D

dashboards
about 181
access, granting 186
content, adding 183
creating 181
list of reports, displaying 185
managing process 177, 181
pages, adding 182
report documents, displaying 185
reports, displaying 184
data
CRM data, extracting with service batch file 159
CRM data, extracting with Siebel ICM 157
data types 49
5,000 records or more, loading into ICM 162
operating unit level data 24
profile attribute data types 49
seed data, configuring CRM to edit 173
10,000 records or more, extracting from CRM 161
Date data 49

debugging information
about 199
logging level detail, adjusting 201
logging level detail, changing 201
logging level detail, reducing 201
logs, generating and sending 202
process of generating 200
display types, extended attributes 62
DSMaxFetchArraySize parameter value, changing 161

E

earning calculation service
about 137
earning summarization service
about 137
Edit privilege 37
editing
extended attributes 64
labels 67
Employee Entity field mappings 159
employee job, modifying 101
Employee objects
core entities 243
employee schema 244
employee.xsd sample XML document 245
sales transactions 246
employee.xsd
recordset 244
sample XML document 245
employees
adding to a region 113
adding to a territory 116
definition 97
importing 39, 136
job history, updating 102
job status, changing 101
setting up 98
enterprise unit
about 23
setting up 26
entities 17
error messages
debugging information 199
log, generating and sending 202
logging level detail, adjusting 201
logging level detail, changing 201
logging level detail, reducing 201
process of generating debugging information 200
errors
See log files
event types 56
exchange rates
importing 45
setting 45
exchange table and table rate
about 268
dependent objects 268
exchange table schema 268
sample XML document 268
exchange.xsd
recordset 268
sample XML document 268
sample XSL document 268
extended attributes
about 61
creating 62
display types 62
editing 64
setup overview 19
extracting 10,000 or more records from CRM 161

F

Failed Attempt Limit password property 121
Failed Attempt Timeframe password property 122
features, new 11, 12, 14
File Adaptor command line utility 242
finalize payment service 137

G

Goal object
core entities and dependent entities 252
Goal XML schema 252
sample Goal XML document 253
Goal XML schema
goal.xsd 252
recordset 252
sample XML document 253

H

Hash Method password property 121
Hdr Field (String) <operator> <value> 215
hierarchies
child regions and territories, adding 117
organization hierarchy levels, adding 88
organization hierarchy, constructing 88
and organizations 87
product hierarchies, adding levels 92
product hierarchy, constructing 92
and products 91
region and territory hierarchy, constructing 112
region hierarchy levels, adding 112

I

ICM calendar 31
ICM configuration
about access 23
ongoing processes 21
roadmap 20
setting up access 24
ICM screen access, adding to responsibilities 173
ICM screen tab objects
creating 168
Siebel application, adding to 173
ICM system administrator
overview of ICM Setup 17
setting up 25
ICM tab, testing in Siebel CRM 175
import services
about 135
import employee service 136
process 135
importing
employees 136
service batch 141
incentive entities 17
incentive types
about 56
setting up 59
Informatica and Publish Service
Publish Service
Informatica setup 130
Informatica mappings
Analytics updates 234
objects for sales transaction data model tables 232
Sales Transaction Bulk Loader 224
sales transaction data model objects 230
source data file formats 225
Transaction Load Module limitations 233
transaction model 223
workflow mappings 234
inherent attributes
about 62
setup overview 19
installing
integration objects into CRM 152
integration workflow 151
interactive report
dashboard, displaying on 184
report documents, displaying on a dashboard 185
reports list, displaying on a dashboard 185
interface
customizing 154
siebelInterfaceConfig 155
workflowConfig 156
invoices
See sales transactions

J

job
history, updating 102
information, modifying 101
job codes
about 79
setting up 80
Job objects
core entities 243
employee schema 244
employee.xsd sample XML document 245
jobs
job codes 79
setting up 80
JProfiler
about 235
profiling guidelines for ICM 236
specifying settings for ICM 235
JVM garbage collector
about 195
optimizing for ICM 196

L

labels
about 67
editing 67
Line Field (String) <operator> <value> 213
Line Product Is Specified Product template 211
Line Rep of Given Rank template 216
Line Rep of Given Rank's Organization at Given Level template 218
loading 5,000 or more records into ICM 162
LocalHost Access log 200
locations
about 80
setting up 84
Lock on Failed Attempt Limit password property 122
Lock Timeout password property 122
log files
about 199
advanced rolling file appender 203
appender definitions 202
appenders and categories 202
Application Log 199
application rolling file appender 203
application server rolling file appender 204

Audit log appender 204
Boot log 200
category definitions 205
Console appender 203
LocalHost Access log 200
logging level detail, adjusting 201
logging level detail, changing 201
logging level detail, reducing 201
logs, generating and sending 202

M

management model
about 17
attributes 19
context 20
entities 17
periods and versions 20
matrix expression
about 268
matrix XML creation notes 271
sample matrix in tabular format 269
sample XML document 270
validations, general 270
XML schema 269

N

Number data 49
Number of Passwords to Remember password property 124

O

obsolete data
removing with the JVM garbage collector 195
removing with the Reaper 189
open integration framework
about 239
basic XML schematics 239
Channel Partner entity 265
Credit object 250
Customer object 255
date and time formats 242
Employee and Job objects 243
exchange table and table rate 268
File Adaptor command line utility 242
Goal object 252
import object schemas 242
matrix expression 268
Organization and Organization Hierarchy 259
Product and Product Hierarchy 253
sorting imported XML file 242
Territory and Territory Hierarchy 260
XML files, profile attributes, and custom attributes in 240
operating unit
about 28
data 24
employees, importing 136
Organization object
about and dependent objects 259
Organization schema 259
sample XML document 260
Organization XML schema
organization.xsd 259
recordset 259
sample XML document 260
organizations
about 87
adding organization hierarchy levels 88
child organizations, adding 90
organization hierarchy, constructing 88
root element, setting up 89
setting up organizations 88

P

page specific labels
about 67
editing 67
parent channel, setting up 104
participants
about 97
channel partners, about 97
channel partners, adding certifications 105
channel partners, adding contacts 105
channel partners, adding details 104
channel partners, creating 103
channel partners, setting up process 103
customers, about 98
customers, adding 106
customers, adding contacts 109
customers, adding details 107
customers, adding locations 108
customers, setting up process 106
definition and types of 97
employee job history, updating 102
employees, about 97
employees, changing job status 101
job history, updating 102
Password Minimum Age password property 123
Password Minimum Length password property 123
password policy
configurable properties 120
filtering log messages 125
setting up 120

Password Strict Syntax password property 123
payroll systems
about 80
setting up 83
performance
DSMaxFetchArraySize parameter value, changing 161
extracting 10,000 or more records from CRM 161
loading 5,000 or more records into ICM 162
performance data
See Credit Object
periods
about 20
calendar years and 31
Position Entity field mappings 161
privilege types 37
processing services
about 136
close period service 138
cumulate service 137
earning calculation service 137
earning summarization service 137
finalize payment service 137
rollup service 137
sales crediting service 136
trial payment calculation service 137
update analytics service 138
product and product hierarchy
core entities and dependent entities 253
product XML schema 253
sample XML document 254
Product Entity field mappings 160
product hierarchy
child product, adding 95
root element, setting up 91
product measures 91
Product object
core entities and dependent entities 253
product XML schema 253
sample XML document 254
product rates and rate groups
about 91
setting up rate groups 92
Product XML schema
about 253
product.xsd 253
recordset 253
sample XML document 254
products
about 91
child products, adding 95
product hierarchy levels, adding 92
product hierarchy, constructing 92
product hierarchy, setting up root element 91
product measures, about and example 91
product rates and rate groups 91
rate groups, associating with the product 94
rate groups, relationship to 91
rate groups, setting up 92
root element, setting up 95
setting up products 93
profile attributes
about 49
aggregate operators 50
data types 49
defining 52
defining process 51
Returnable fields 51
Reversible fields 51
profiles
about 49, 51
defining process 51
setting up 52
profiling
about JProfiler and ICM 235
ICM profiling with JProfiler guidelines 236
specifying JProfiler settings for ICM 235
Publish Service
about 129
configuration file 130
Informatica properties, editing 130
process 129
running in a service batch 133
running standalone 133
service batch item 134
Purge Period Data service 189

Q

qualifiers
about 111
adding to a territory 115

R

rate groups
jobs, relationship to 79
product rates and 91
products and 91
products, associating with 94
setting up 92
Reaper
about 189
configuring 189
recipient templates
about 74

examples 216
Line Rep of Given Rank 216
Line Rep of Given Rank's Organization at Given Level 218
setting up 75
reference data
about 79, 97
channel segments, about 79
channel segments, setting up 84
cost centers, about 79
cost centers, setting up 83
job codes, about 79
jobs, setting up 80
locations, about 80
locations, setting up 84
payroll systems, about 80
payroll systems, setting up 83
salary grades, about 80
salary grades, setting up 82
setup tasks 80
reference implementation field mappings
Account Entity field mappings 159
Employee Entity field mappings 159
Position Entity field mappings 161
Product Entity field mappings 160
regions
about 111
adding channel partners 114
adding child regions to a hierarchy 117
adding employees 113
adding region hierarchy levels 112
region and territory hierarchy, constructing 112
setting up 113
Relative period 31
release, new features 11, 12, 14
report documents
about 177
adding an ICM folder on the Actuate server 178
defining security access 179
displaying on a dashboard 185
granting access to the Actuate server 178
making available in ICM 179
setting up process 177
reports
displaying on a dashboard 184
list, displaying on a dashboard 185
Returnable fields 51
Reversible fields 51
Revoke privilege 37
roles
assigning 40
new roles, adding 39
privilege types 37
security and 37, 119
users 37
rollup service
about 137
root element
of organization hierarchy, setting up 89
of product hierarchy, setting up 91, 95

S

Safe Modification password property 124
salary grades
about 80
setting up 82
sales crediting service
about 136
Sales Transaction Bulk Loader, mapping process 224
sales transaction data model
objects 230
tables 232
Sales Transaction XML schema
record set 248
salesTrans.xsd 248
sample XML document 249
sales transactions
adjustments 247
core entities 246
data model comments 246
Extended Attributes and 246
salesTrans.xsd sample XML document 249
XML schema 248
Search privilege 37
security
adding users 40
assigning roles 40
importing employees, customers, and channel partners 39
new roles, adding 39
password policy, configurable properties 120
password policy, filtering log messages 125
password policy, setting up 120
privilege types 37
roles 37, 119
setting up process 38
users 37
seed data, configuring CRM to edit 173
segment types 32
service batch
about 139
creating 140
importing 141
process of creating and managing 140

Publish Service 133
schema 273
Service Batch Framework 139
Service Launcher properties 142
Service Launcher, about 139
Service Batch Framework 139
Service Launcher
about 139
properties 142
properties file, setting up 142
service batch file, Publish Service item 134
service batch file, setting up 146
setting up process 141
services
about 135
export services 136
logging level, setting 138
services, export
about 136
Publish Service 129
services, import
about 135
import employee service 136
process 135
Publish Service 129
services, processing
about 136
close period service 138
cumulate service 137
earning calculation service 137
earning summarization service 137
finalize payment service 137
rollup service 137
sales crediting service 136
trial payment calculation service 137
update analytics service 138
setup entities 17
Siebel CRM-to-ICM integration
about 149
Account Entity field mappings 159
connect string, testing 163
CRM data, extracting with service batch file 159
CRM data, extracting with Siebel ICM 157
CRM extract, loading into ICM 159
data loads into ICM 150
DSMaxFetchArraySize parameter value, changing 161
Employee Entity field mappings 159
extract/transform/load pattern 149
5,000 or more records, loading into ICM 162
full and incremental extracts 150
ICM configuration properties, setting 153
ICM interface, customizing 154
integration objects, installing 152
integration workflow, installing 151
Position Entity field mappings 161
process of running 151
Product Entity field mappings 160
reference implementation entities 150
search specification string 150
10,000 or more records, extracting from CRM 161
troubleshooting 162
Siebel CRM-to-ICM single sign-on
See single sign-on
Siebel ICM Service Batch XML 235, 273
single sign-on
about 165
configuring, process 165
CRM responsibilities, identifying for ICM access 166
CRM, configuring to edit seed data 173
ICM properties, setting for CRM authentication 166
ICM roles, mapping to CRM responsibilities 167
ICM screen access, adding to CRM responsibilities 173
ICM screen access, adding to responsibilities 173
ICM screen tab objects, creating 168
ICM screen tab objects, creating in CRM 168
ICM screen tab, adding to Siebel application 173
ICM tab in Siebel CRM, testing 175
ICM Web templates, copying to CRM 168
smart attribute scripts
about 209
Customer Code smart attribute 209
smart attributes
about 69
defining 70
standard calendar years
custom and standard calendars, combining 32
setting up 33
String data 49
summarized earning 137
supervisor, associating with employees 136
system administrator
ICM system administrator, overview ICM Setup 17
ICM system administrator, setting up 25
system entities 17

T

template scripts
about 211
condition template examples 211
recipient template examples 216
templates
about 73
condition templates, about 73
condition templates, setting up 74
copying ICM Web templates to CRM 168
Hdr Field (String) <operator> <value> 215
Line Field (String) <operator> <value> 213
Line Product Is Specified Product 211
Line Rep of Given Rank 216
Line Rep of Given Rank's Organization at Given Level 218
recipient templates, about 74
recipient templates, setting up 75
territories
about 111
channel partners, adding 116
child territories, adding to a hierarchy 117
employees, adding 116
qualifiers, about 111
qualifiers, adding 115
region and territory hierarchy, constructing 112
regions, setting up 113
root element, setting up 117
setting up 115
Territory objects
about 260
core entities 261
dependent entities 261
sample territory XML document 264
territory schema 262
Territory XML schema
recordset 262
sample XML document 264
territory.xsd 262
testing ICM tab in Siebel CRM 175
transaction entities 17
Transaction Load Module limitations 233
transaction types
about 55
incentive types, about 56
incentive types, setting up 59
setting up process 56
transaction event types, setting up 56, 58
transaction line types, setting up 56, 57
transaction types, setting up 55, 57
trial payment calculation service 137
troubleshooting
connect string, testing 163
debugging information 199
logging level detail, adjusting 201
logging level detail, changing 201
logging level detail, reducing 201
logs, generating and sending 202
process of generating debugging information 200
Siebel CRM-to-ICM integration 162
types
See transaction types

U

unit currencies, assigning 44
update analytics service 138
users
about 37
adding 40

V

versions 20
View privilege 37

X

XML
basic pattern schematics example 239
date and time formats 242
import object schemas 242
profile attributes and custom attributes 240
service batch schema 273
sorting imported XML file 242
XML documents
See open integration framework
XML service batch schema 235, 273