Ericsson Order Care

Realize Higher Consistency for Faster Time-to-Revenue

Customer Information Management Configuration Guide

2/127 01-CSH 109 420/1 Uen Rev A

Contents

1 Introduction

This document provides information on configuring the Customer Information Management (CIM) module.

1.1 Purpose and Scope

This guide provides the reader with an understanding of the data model and features supported by the Customer Information Management (CIM) module. To perform the tasks in this document requires that you have metadata development experience.

1.2 Reader’s Guideline

This section describes the version syntax covered in this document and any additional, required information.

Commands that you enter on the command line appear in courier font, such as the following:

```
svnadmin dump C:\SVN\myProject > C:\backupFolder\myProject.bak
```

Document names and sections within documentation are set in italics, such as the following:

For more information on making a copy of your project metadata, see the *Velocity Studio User Guide*, under *Velocity Studio User Interface* > *Common Actions Outside Velocity Studio*.

Note: To navigate the documentation, an arrow appears (>), which separates each hyperlink to be clicked.

2 CIM Overview

The Customer Information Management (CIM) module is an application for managing Customers, Billing Accounts, Addresses, and Customer contacts. The module is aligned with the Telecom Shared Information/Data Model (SID) and therefore implements concepts such as Parties and Party roles.

The CIM module provides the following features:

Search customer information

Search functionality available to find an individual customer or organizational customers.

Manage customer information

A new customer can be created or an existing customer's information can be modified. Customer information such as Site or Address (billing or service addresses), contact information (phone numbers, email addresses), customer contacts, notes, and so on can easily be searched, added, deleted or modified.

Customer account

Customer account information can be accessed to view the account profile information. Account status (New, Active, Blocked and Inactive) indicates the state of the account and contains business rules associated with the status.

360° view of the customer

The CIM application offers a 360° view of the customer information in a hierarchical relationship. Multiple customer contacts, addresses, accounts, services, orders and history can be viewed and accessed from a single tree node menu.

Integration

CIM is fully integrated with other Ericsson applications such as Order Management, Order Negotiations and Service Registry. Viewing the customer information allows a view of the quoted orders, orders being provisioned and existing customer services.

Search and drill down capabilities

The CIM application offers various finder forms that allows for information to be readily retrieved. From each finder form, the details of the results rows can be clicked and the details of that row are displayed.

External application integration

The API functionality allows for the integration of CIM to other external applications such as Billing and Payment systems. The External Identifiers feature integrates the Customer ID from the external system to the customer record in the CIM application.

3 Quick Start

The following are the quick steps to install the CIM module:

1. Install Velocity Studio and follow the directions from the Velocity Studio’s Installer User Guide to install Velocity Studio, initialize the database, and configure the System Configuration application.
2. Create a project in Velocity Studio and set the internal name of the project.
3. Add the following library files for CIM module.

4. Start the runtime and update the logical connection in the System Configuration application.
5. Upgrade the database and run the associated SQL file.
6. Assign the privileges required to run the CIM module in the User Profile Management application.
7. Logout and log back in from the runtime.
8. Double-click the Customer Information Management application’s icon.

4 Installation and Setup of CIM

This section provides the details on how to install and configure CIM module.

4.1 Set up Velocity Studio and Database Schema

1 Install Velocity Studio as outlined in the *Installer User Guide > Standard Install.*

2 Create a new schema in your database. Refer to the *Installer User Guide > Standard Install > Database Initialization.*

3 Run the <*EOC-ECM_installation_folder*>\DDL\CW.sql file. Use your newly created database schema to run this script. Update the `CW.sql` with the new user name.

4 Open the <*EOC-ECM_installation_folder*>\designer\env\startDesigner.cmd file to start Velocity Studio.

5 Create a Project in Velocity Studio.

  a Click **File > New > New Project** from the menu bar.
  b From the Select an empty directory dialog, specify the folder where you want to save your new project.
  c Click the root metadata node, by default appears as AVM Metadata. On the General properties of this node, enter project’s internal name in the **Internal Name** field.

6 Click **Database > Connect** from the menu bar to connect to your newly created database schema.

7 From the Database Login dialog, click the **New** button to configure the database connection settings.

8 The Connection Properties dialog appears; enter the name for the connection and the name of your database schema user in the **Name** and **User** fields, respectively.

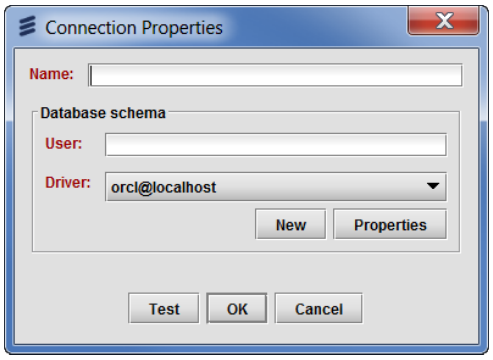


Figure 1 Connection Properties Dialog

9 Click the **New** button.

10 The Driver Properties dialog appears; click the **Driver type** field's drop-down menu and select **Oracle thin**. Proceed to enter the **Host, Port, Connection Type,** and **Service** information.

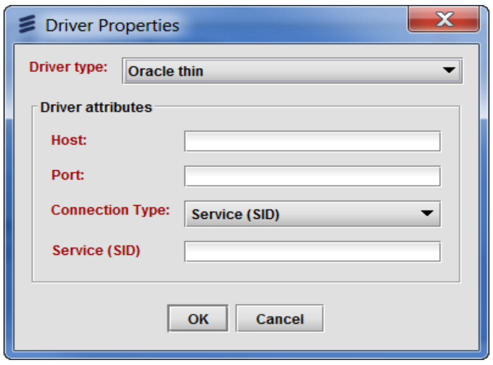


Figure 2 Driver Properties Dialog

11 Click the **OK** button.

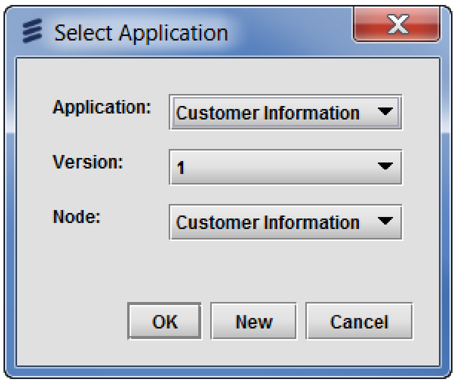
12 The Connection Properties dialog reappears. You can click the **Test** button and then enter the password to test the connection.

13 Click the **OK** button to return to the Database Login dialog.

14 Enter the value in the **Password** field and click the **OK** button to connect.

15 Click **Runtime** > **Run** from the menu bar to run the framework.

16 The Select Application dialog appears; click the **New** button.


Select Application

Application: Customer Information

Version: 1

Node: Customer Information

Figure 3 Select Application Dialog

17 The Add dialog appears; enter the value for the **Version** and **Description** field.

18 Click the **OK** button from the **Add** and **Select Application** dialogs.

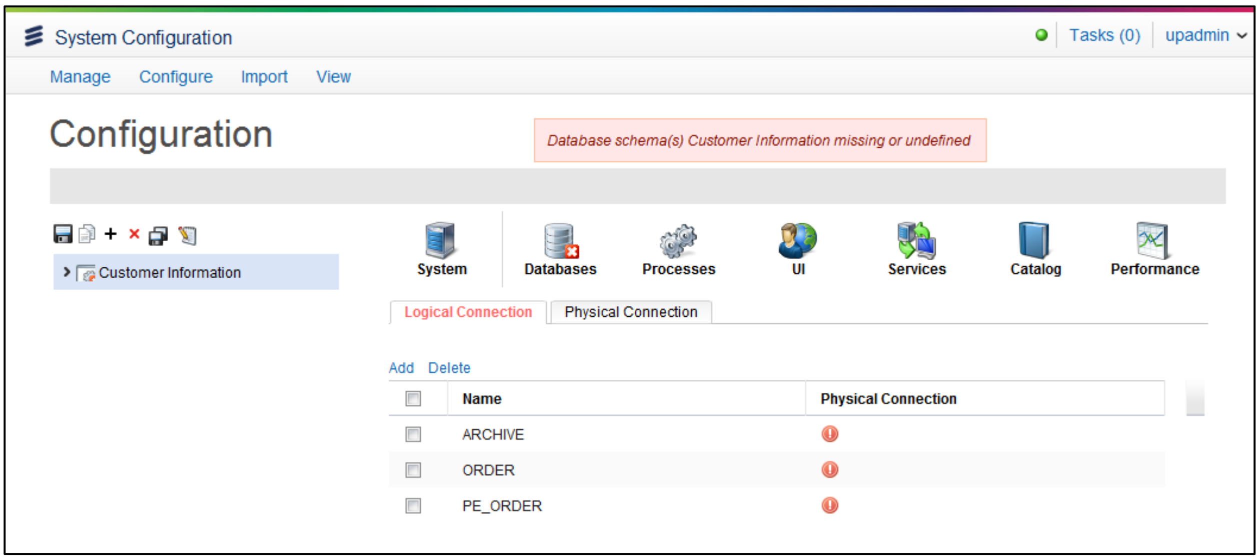
Note: The Velocity Studio runs in Configuration mode until the System Configuration application is properly set up, and the application metadata has been run.

4.2 Set Up the System Configuration Application

The following are the steps to set up the system configuration application.

1 In your Web browser, access the System Configuration application by entering `http://localhost:8080/cwf/config` as the URL.

2 Enter **updadmin** as both your Username and Password, and then press the **Enter** key to login. The main screen of system configuration application displays:



Configuration

Database schema(s) Customer Information missing or undefined

Name	Physical Connection
ARCHIVE	●
ORDER	●
PE_ORDER	●

Figure 4 Main Screen of System Configuration Application

3 Select the main node (for example, Customer Information) from the node menu section, and then click the **Database** tab.

4 The **Logical Connection** tab displays the logical connections to the database, which are responsible for enabling the AVM to send database commands to the database, to carry out different functionalities.

5 Clicking **Databases** > **Physical Connection** displays the physical connections to the database, which are responsible for defining database connection parameters to be used by logical connections. Complete these steps to add a physical connection:

   a. From the Physical Connection page, click the **Add** button.
   b. The New Physical Connection dialog appears. Enter your database credentials.
   c. To test your connection settings, click the T**est** button. If your connection settings are properly set up, a *Successful Connection* confirmation message appears.
   d. Click the **Apply** button. A message appears, indicating that you have successfully updated your Oracle thin connection.
   e. Click the **Close** button.

Note: The database attributes in the System Configuration application need to match the database attributes in the Velocity Studio. For more information, refer to the *System Configuration User Guide*.

6 Click the Logical Connection tab to associate your logical connections to the physical connection you have just created, double-click each of the following logical connections and select your newly created physical connection from the drop-down list:

- ARCHIVE
- ORDER
- PE_ORDER

7 Click the **Save** button to save your configuration settings, and exit the system configuration application.

Note: You can select the **Active configuration** checkbox for **PE**, **PE_UI**, or **UI** node.

4.3 Add JAR files

Go back to the Velocity studio, and follow these steps to continue the configuration on the Velocity Studio side.

1 Click **Runtime > Stop** from the menu bar, to stop the runtime in Velocity Studio.

2 Click the root metadata icon (for example, AVM Metadata) in the left navigation menu, and then click the **Library** tab.

3 Click the Add button () to launch the Select an template JAR dialog.

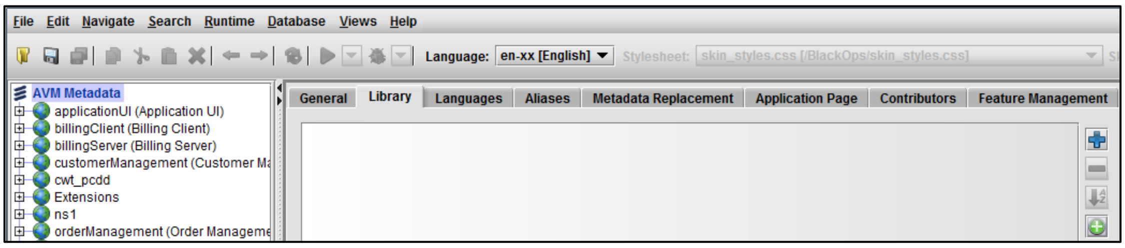

File Edit Navigate Search Runtime Database Views Help
Language: en-xx [English]
Stylesheet: skin_styles.css [/BlackOps/skin_styles.css]
AVM Metadata
applicationUI (Application UI)
billingClient (Billing Client)
billingServer (Billing Server)
customerManagement (Customer Ma
cwt_pcdd
Extensions
ns1
orderManagement (Order Manageme
General
Library
Languages
Aliases
Metadata Replacement
Application Page
Contributors
Feature Management

Figure 5 Add Library Files

4 Select the following JAR files required for CIM module from <*EOC-ECM_installation_folder*>\modules folder:

Module	Required JAR Files	Recursive JAR Files
Customer Information Management	• customer_information_managment.jar • catalogClient.jar	• address.jar • api_common.jar • billing.jar • customer.jar • cwl_address.jar • cwl_customer.jar • cwl_party.jar • data_dictionary.jar • notification.jar • party.jar

5 A Copy File dialog appears; select Yes or No in the dialog. If Yes is selected, the JAR files are copied locally to your <*EOC-ECM_installation_folder*>\templates folder. If No is selected, file path is added to your template folder.

6 Once the files are added, save the project metadata.

7 Reload or open the project for the library files to take effect.

8 Run the following SQL file to create the CIM privileges:

<EOC-ECM_installation_folder>\modules\customer_information_management\DDL\customer_information_management.sql.

9 Click **Runtime** > **Run** from menu bar to start the runtime. The Velocity Studio runs in configuration mode. New logical connections, **CATALOG** and **CUSTOMER**, are available in the System Configuration application.

10 Follow the steps described in the previous section of this document to login to the System Configuration application, and to associate new logical connection to the physical connection.

11 Click the **Save** button and exit the System Configuration application.

12 To make the existing database schema compatible with the new files and settings, upgrade the database by following these steps:

   a. Select **Database > Upgrade System** from the menu bar to open the Upgrade SQL dialog.
   b. Specify the directory and the name of the SQL file (for example, upgrade.sql), and then click the **Save** button to create the file.
   c. Use SQLPlus or SQL Developer to connect to the appropriate database and run this upgrade file.

Note: If there are no system upgrades available, a dialog box appears, indicating that no upgrades are required.

4.4 Assign the Privileges

To assign the privileges, complete the following steps:

1. In the Velocity Studio, click **Runtime > Run** from the menu bar or click the run button (▶) to start the framework.

2. Open your Web browser and enter the `http://<localhost>:<port>/cwf/login` Web address. For example, `http://localhost:8080/cwf/login`.

3. Enter the username and password to login (for example, `upadmin` for both the **Username** and **Password** fields), and then click the **OK** button.

4. Open the User Profile Management application and click the **Manage > Groups** from the menu bar.

5. On the Search Group page, click on the **Search** button to get the list of the user groups.

6. Double-click the appropriate user group (for example, User Profile Administrators).

7. On the Select Privileges page, first click the **Edit** button and then click the **Add** button. The Search Privileges page appears with the available privileges for that group.

8. Select all privileges and click the **Select** button. A message appears that the privilege has been added successfully.

9. Click **Upadmin** option from the menu bar, and then click the **Logout** option.

10. Log back in to the application; follow the steps defined previously in this section.

11. The **Application Selection** page appears with the available applications.

12. Double-click the icon for Customer Information Management module. The main screen of the application appears as follows:

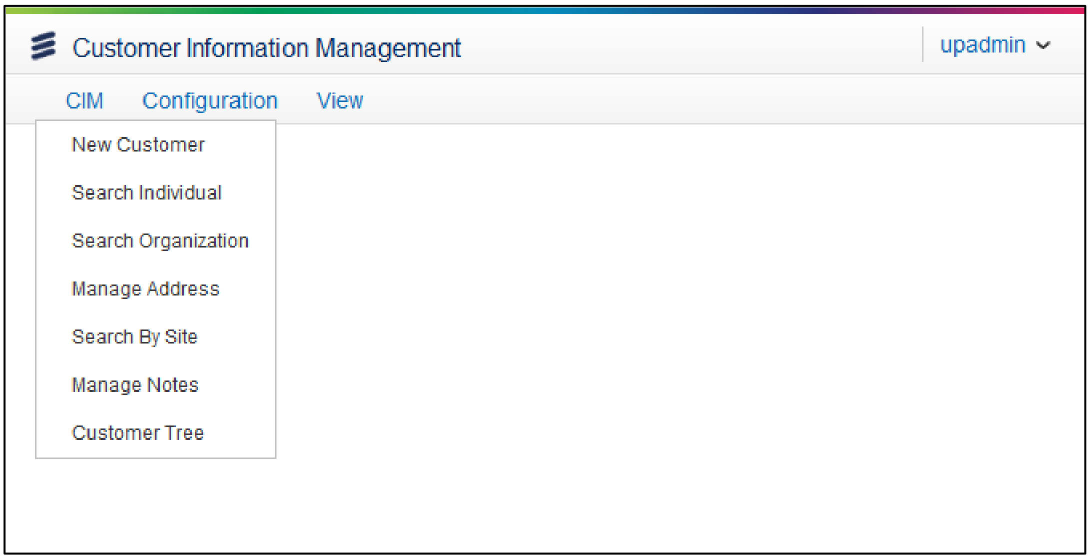


Figure 6 CIM User Interface

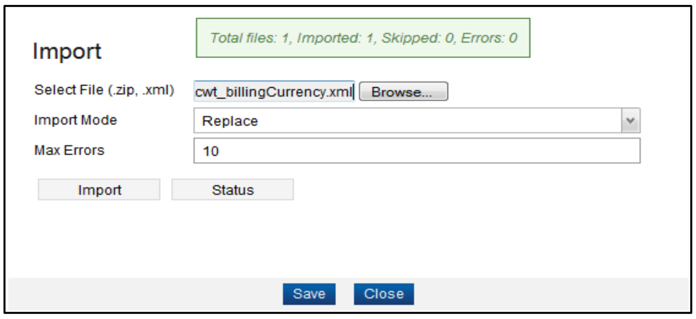
Notes:

- If the login screen does not load, verify that either the Web address is correct or contact your system administrator to verify that you have the correct Web address.
- The CIM application does not contain any data. To use this application, you need to import the data from code tables.

5 Working with Code Tables

To get the default data for CIM, you need to import the code tables. The CIM related code tables are available in the <*installation_folder*>\modules\customer_information_management\code_tables folder. To Import the code tables, follow these steps:

1. From the menu bar of CIM application, click **Configuration** > **Code Tables** option.
2. The Code Tables page appears; click the **Import** button from the results section.
3. The Import dialog appears; the following table describes the fields for this dialog.
4. Click the **Browse** button and select a code table (for example, accountStatusCT.xml) from <*installation_folder*>\modules\customer_information_management\code_tables folder.
5. Click the **Import** button. A confirmation message appears with the summary.


Total files: 1, Imported: 1, Skipped: 0, Errors: 0

Import

Select File (.zip, .xml)	cwt_billingCurrency.xml
Import Mode	Replace
Max Errors	10

Figure 7 Import Code Tables

The CIM module depends on the following code tables:

Code Table	Label	Description	Values (example)
cwt_addressRole	Addresses Roles	address roles for a postal address contact medium	Business Address
cwt_contactMediumType	Types of Contact Mediums	The type of the contact medium	phone, mobile, fax, email, postal addr, pager, url, etc
cwt_contactTypeValidation	Contact Medium Types – regexp validations	The validation for the contact medium type (regexp)	\d+
Industry	Industry	Types of industries (organizations)	TEL - telecom
cwt_IndIdentificationType	Types of IDs for Individuals	Types of IDs for Individuals	Driver's License, Passport, Security Questions
cwt_OrgIdentificationType	Types of IDs for Organizations	Types of IDs for Organizations	Registration number
cwt_OrgNameType	Organizations Trading Name Types	Organizations Trading Name Types	Co, Gmbh, Inc, Ltd, Plc, Pty
cwt_party_PartyRole	Party Role - Type	Types of roles that a party can play	Account Contact, Customer, Customer Contact,

Code Table	Label	Description	Values (example)
			Site Contact
provState_<COUNTRYCODE>	Provinces & States by Country		
iso3166	ISO 3166 Country Codes	Country codes	CA, US, etc
cwt_ContactNameType_Ind	Types of Names for Individual	Types of Names for Individual	Maiden Name, Married Name, etc
cwt_ContactNameType_Org	Name Types for Organization	Name Types for Organization	Short Name, Operating Name, etc
cwt_creditCardType	Types of Credit Card	Type of Credit Card	Visa, Master, etc
cwt_externalSystem	External Systems	Application defined External System names	
marketseg	Market Segment	Customer market segment	Residential, Government , etc
iso6392	Language Codes (3 char)		eng, por
iso6391	Language Codes (2 char)		en, pt
iso4217	Currency Codes		CAD, BRL
cwt_billingCurrency	Customer Account Billing Currency		CAD, USD
cwt_invoiceLanguage	Customer Account Invoice Language		English, French
cwt_dictInvoiceOptions	Customer Account Invoice Option		Email, mail
cwt_siteResponsibility	Site Responsibility	Site Contact specific role for the site (technical support, etc).	Technical support
cwt_noteSubType	Note Sub Type	Note types codes	Other

Code Table	Label	Description	Values (example)
accountStatusCT	Account Status	Account Status Codes	New, Active, Inactive
customerStatusCT	Customer Status	Customer Status Codes	Prospect, Inactive, Active, Blocked
cwt_creditCheckResult		CIM Credit Check Result	Credit Failed, Credit Passed
cwt_customerSubType	Customer sub Type	Customer Sub Type	Enterprise, Carrier, Wholesale, VIP, Employee
cwt_customerSubType_Commercial	Customer Sub Type – Commercial	Customer Sub Type – Commercial	Enterprise, Carrier, Wholesale
cwt_customerSubType_Residential	Customer Sub Type – Residential	Customer Sub Type – Residential	VIP, Employee
cwt_customerType	Customer Type	Customer Type	Commercial, Residential
cwt_noteType	Note type	Note type codes	Task, Phone, Email, Letter, Appointment, Other
cwt_orderStatus	Order Status	Order Status codes	Draft, Config
cwt_organizationType	Organization Type	Organization Type codes	Sole Proprietor, Corporation, Partnership, S-Corporation, Trust, Non-profit organization
cwt_SecurityQuestions.xml	SecurityQuestions	Security questions codes	QUESTION1, QUESTION2, …

6 Initialize Configuration

Configuration data is stored in both code tables and regular document tables. The contents of code tables should be reviewed and updated as required. Refer to the Code Table section of the CIM User Guide for further details.

Additional configuration information needs to be reviewed and updated. This is accomplished by running the CIM application and accessing the Configuration menus.

6.1 Address management > Configuration >Types Of Address per Country

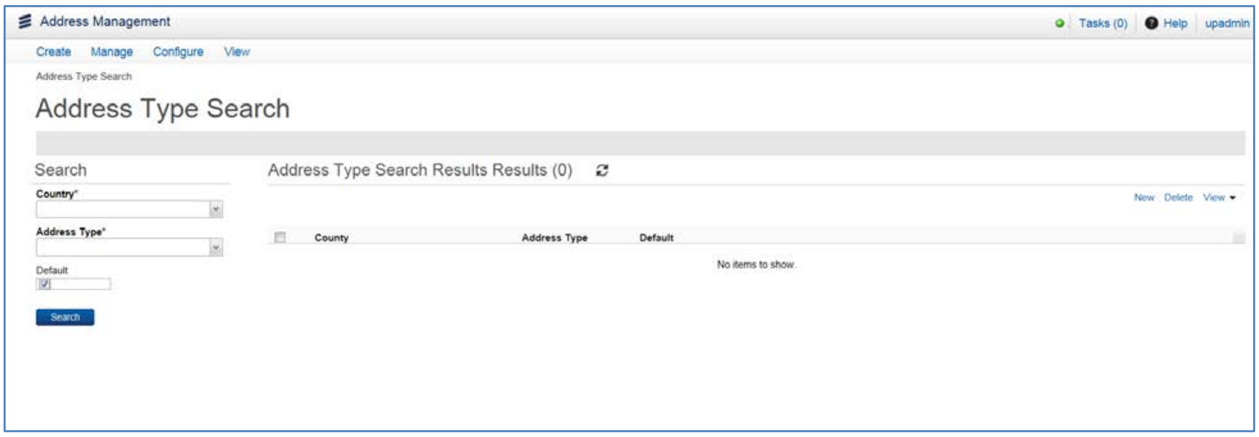
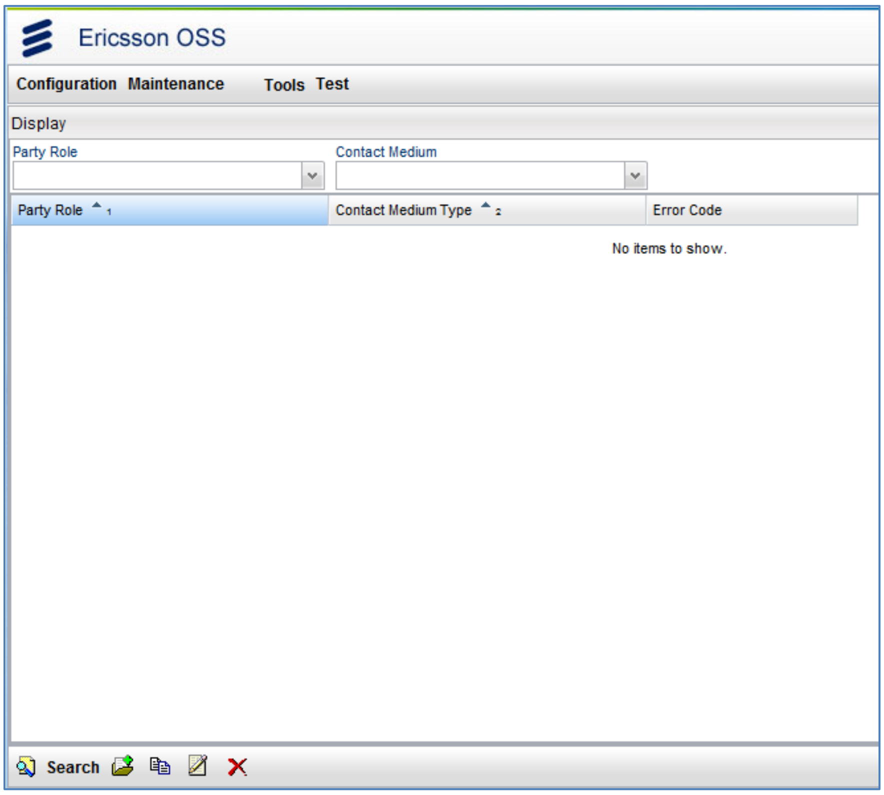


Figure 8 Types of address per country

Script Name	Description
cwtAddress.formatAddress	Used to format an address based on country code.
cwtAddress.formatAddress_CA	Used to format a Canadian address.
cwtAddress.visualKeyAddressMasterDoc	Visual key.
cwtAddress.visualKeyCityDoc	Visual key.
cwtAddress. visualKeyExternalAddressIdentifier	Visual key.
cwtAddress. visualKeyRegionDoc	Visual key.
cwt_party.isPartyNotUnique	Used to determine whether a given party record is unique.
cwt_party.validateParty	Receives a party document and checks if it is consistent. The script is called from a validation rule in the Party Document. It should return ‘true’ if the document is invalid. The provided script checks if the Party has a primary name.

Script Name	Description
cwt_party.visualKey*	Visual key.
cwt_party.partyRoleCustomer	Returns the party role type which represents a customer.
cwt_cust.validateCustomer	Validates a customer entity.
cwt_cust._visualKeyAccount	Account Node Visual Key
cwt_cust._visualKeyContact	Customer Contact Node Visual Key
cwt_cust._visualKeyCustomer	Customer Node Visual Key
cwt_cust._visualKeyNote	Note Visual Key
cwt_cust._visualKeyQuote	Quote Node Visual Key
cwt_cust._visualKeySite	Site Node Visual Key
cwt_cim. _newQuote	Should be overridden by ON to create a new quote
cwt_cim._openQuote	Should be overridden by ON to open a new quote
cwt_cim._quoteFinderSelect	Should be overridden by ON to search for a quote
cwt_cim._getMapUrl	Should be overridden by Application to customize address format.

6.2 Party Management > Configuration > Mandatory Contact Medium

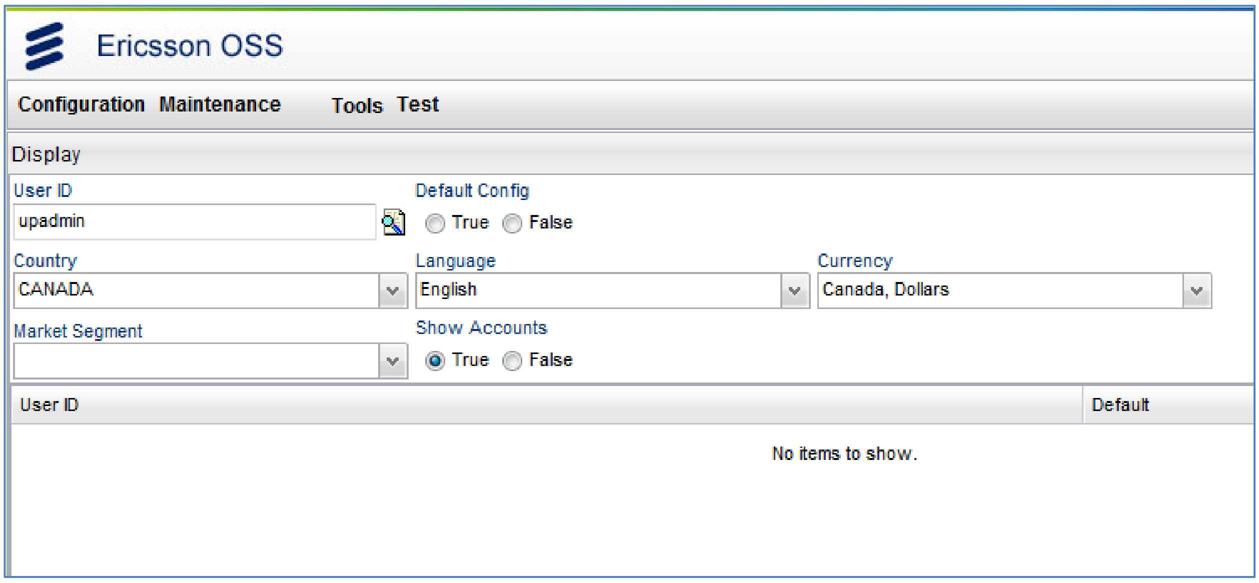


Party Role	Contact Medium Type	Error Code

No items to show.

Figure 9 Mandatory contact Medium

6.3 Customer Management > Configuration > Default Settings



User ID: upadmin

Default Config: ( ) True ( ) False

Country: CANADA

Language: English

Currency: Canada, Dollars

Market Segment:

Show Accounts: [✓ True] ( ) False

User ID	Default
No items to show.	

Figure 10 Default Settings

6.4 Customer Information Management > Configuration > CIM Config

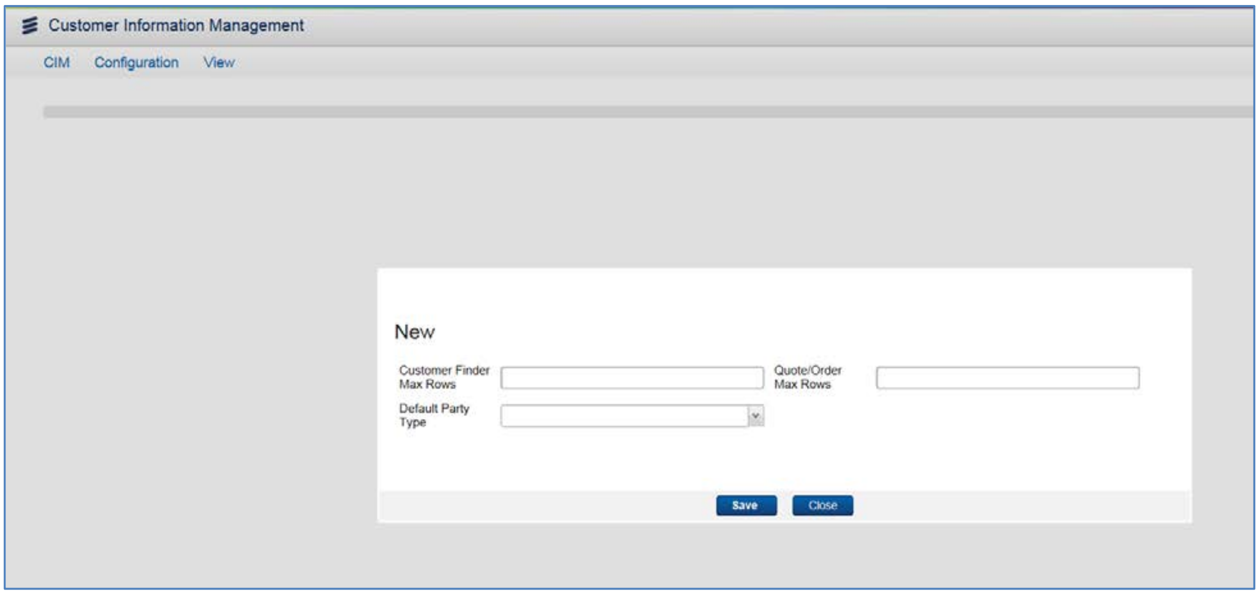


Figure 11 CIM Config

Note: Override scripts. Some scripts defined within the module represent stubs or event handlers that are to be overridden. Verify view permissions.

7 Extending CIM Documents

The steps below describe the process for adding new leaves to CIM Documents.

- Extend and override the desired document (e.g. cwt_customer:customerDoc document), adding the new leaves
- Map the new leaves to DB to the same table used by the customerDoc document. Do not map to a new table.
- Extend and override the conversion Maps used to convert data from Document to Data Structure and vice-versa. (e.g. cwt_customer:customerDoc2customerDS and cwt_customer:customerDS2customerDoc)
    - After creating the new conversion maps, remove the "Map conversion" action scripts that are automatically created by the product
    - All leaves (except the primary key from Data Structure to Document) should be mapped from Data Structure to Document and vice-versa. Use the **Map All** button. After mapping all the leaves, remove the mapping for the primary key only in the conversion map from Data Structure to Document.

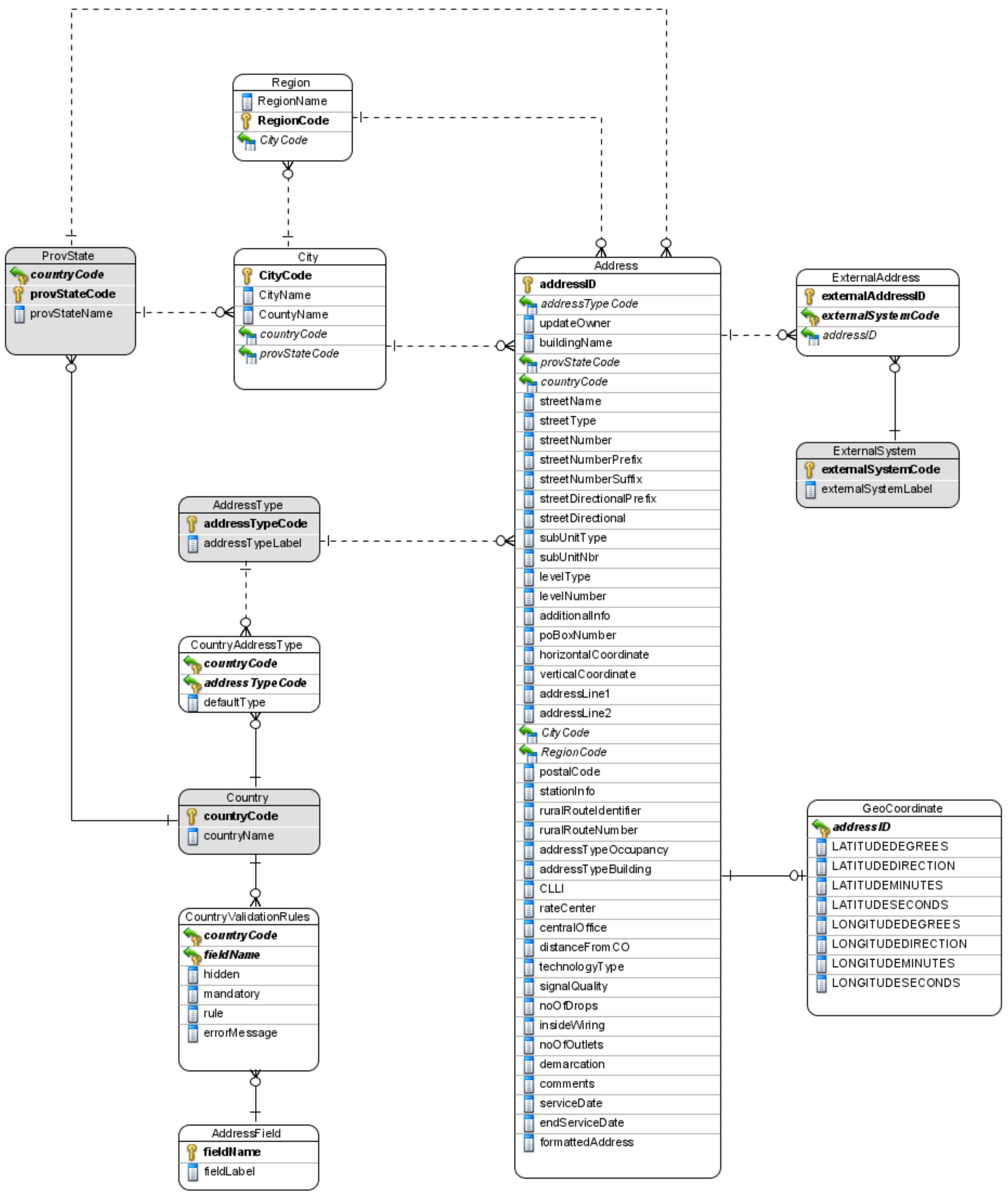
8 Data Model

This section includes entity relationship diagrams and table and field descriptions to provide the reader with an understanding of the entities and their relationships.

Notes:

- Table and column names are provided in camel case.
- Some common fields such as LASTUPDATEDDATE and CWCREATEDBY have been omitted.
- Entity diagrams also include code tables which are represented by entities with a grey background. In Velocity Studio, all code tables are stored in the Oracle table named ‘CWDBCODETABLES’ and not in a table as may be implied by the diagram.

8.1 Addresses Entity



8.1.1 Address

The Address Master document implements the concept of a Geographic Place, as defined by the SID model. An address is always linked through a Contact Medium. When the concept of master addresses is being used, an address can be shared by several contact mediums. The Address Master document also keeps record of the facilities present in each address.

Different types of addresses are supported by the modules, such as Civic Address, Rural Route, PO BOX, Latitude & Longitude, V&H Coordinates and General Delivery.

Each type of address has its own form and every field displayed in each form can be made visible or invisible, mandatory or optional and can have validation rules added. All rules are configured per country. The supported types of addresses can also be restricted per country. Also, a default address type can be defined for each country. When editing an address and country is selected, the default address type form for that country is displayed.

Attribute/Column Name	**Type**	**Description**
(PK)addressID	string(32)	Unique identifier.
(FK)addressTypeCode	string(16)	A code used to classify an address. Depending upon the value, different address forms are displayed.
updateOwner	number(1)	
buildingName	string(30)	Name of building
(FK)provStateCode	string(2)	Province/State.
(FK)countryCode	string(2)	Country.
streetName	string(50)	Name of street
streetType	string(10)	
streetNumber	string(7)	House or Unit number
streetNumberPrefix	string(6)	For example, Apt, Unit, etc.
streetNumberSuffix	string(6)	
streetDirectionalPrefix	string(10)	
streetDirectional	string(10)	
subUnitType	string(16)	The type of subunit For example, BERTH, FLAT, PIER, SUITE, SHOP, TOWER, UNIT, WHARF
subUnitNr	string(16)	
levelType	string(16)	Floor, Basement, Penthouse.
levelNumber	string(6)	
additionalInfo	string(50)	
latitudeDegrees	number(3)	Latitude … Degrees, Minutes, Seconds
longitudeDegrees	number(3)	Longtitude … Degrees, Minutes, Seconds
poBoxNr	string(16)	PO Box Number
addressTypeOccupancy	string(6)	Code table:

Attribute/Column Name	Type	Description
		cwtdictOccupancyType
addressTypeBuilding	string(1)	Enumeration. R- residential, A – Apartment …
CLLI	string(32)	Common Language Location Identifier
rateCenter	string(10)	
centralOffice	string(10)	
distanceFromCO	number(5)	
technologyType	string(10)	Code Table: cwtdictTechnologyType
signalQuality	number(3)	
noOfDrops	number(3)	
insideWiring	number(1)	
noOfOutlets	number(3)	
demarcation	string(100)	
comments	string(100)	
serviceDate	date	
endServiceDate	date	
formattedAddress	string(128)	

8.1.2 AddressType

Description: An enumeration. Values include:

CIVIC	Civic Address
RURAL	Rural Route
POBOX	PO BOX
LATLONG	Latitude/Longitude
VH	V&H Coordinates
GENERAL	General Delivery
OTHER	Other

Attribute/Column Name	Type	Description
(PK)addressTypeCode	string(16)	Unique identifier
addressTypeLabel	string(128)	

8.1.3 Country

Description: A code table of countries. ISO 3166 Country Codes.

Attribute/Column Name	Type	Description
(PK)countryCode	string(2)	Unique identifier.
countryName	string(128)	

8.1.4 CountryAddressType (addrTypeDoc) / CWT_ADDRTYPE

Description: Defines valid addresses types per country. This table is maintained using the Configuration Menu. ISO 3166.

Attribute/Column Name	Type	Description
(PFK)addressTypeCode	string(16)	FK to Address Type
(PFK)countryCode	string(2)	FK to country code
defaultType	number(1)	If true, indicates that this is the default address type for this country.

8.1.5 ProvState_XX

States are defined by code tables, per country. The state code tables' names should be prefixed by 'provState_' followed by the two letters ISO Country Code (e.g., provState_CA for Canada). The codes present in the state code tables must be the official codes for the state (e.g., ON for Ontario), since they will be used in the formatted address, when required.

Attribute/Column Name	Type	Description
(PK)provStateCode	string(16)	Unique identifier
(PFK)countryCode	string(16)	
provStatename	string(128)	

8.1.6 City

Define valid cities by Province\State.

Attribute/Column Name	Type	Description
(PK)cityCode	string(16)	City identifier
cityName	string(64)	Name of the city.
(FK)provStateCode	string(16)	
(PK)countryCode	string(16)	

8.1.7 Region

Define regions within a city boundary.

Attribute/Column Name	Type	Description
(PK)regionCode	string(16)	Region identifier
regionName	string(64)	Name of the region.
(FK)cityCode	string(16)	The city for which the region is defined.

8.1.8 ExternalSystem

Code table representing external systems.

Attribute/Column Name	Type	Description
(PK)externalSystemCode	string(16)	Unique identifier
externalSystemLabel	string(64)	

8.1.9 ExternalAddress

Cross reference table used to associate an address with the id of the address in another system/application.

Attribute/Column Name	Type	Description
(PFK)externalSystemCode	string(16)	
(PK)externalAddressID	string(64)	
(FK)addressID	string(32)	

8.1.10 FieldName

Code table of field names found on the address entity.

Attribute/Column Name	Type	Description
(PK)fieldNameCode	string(16)	Unique identifier.
fieldName	string(64)	

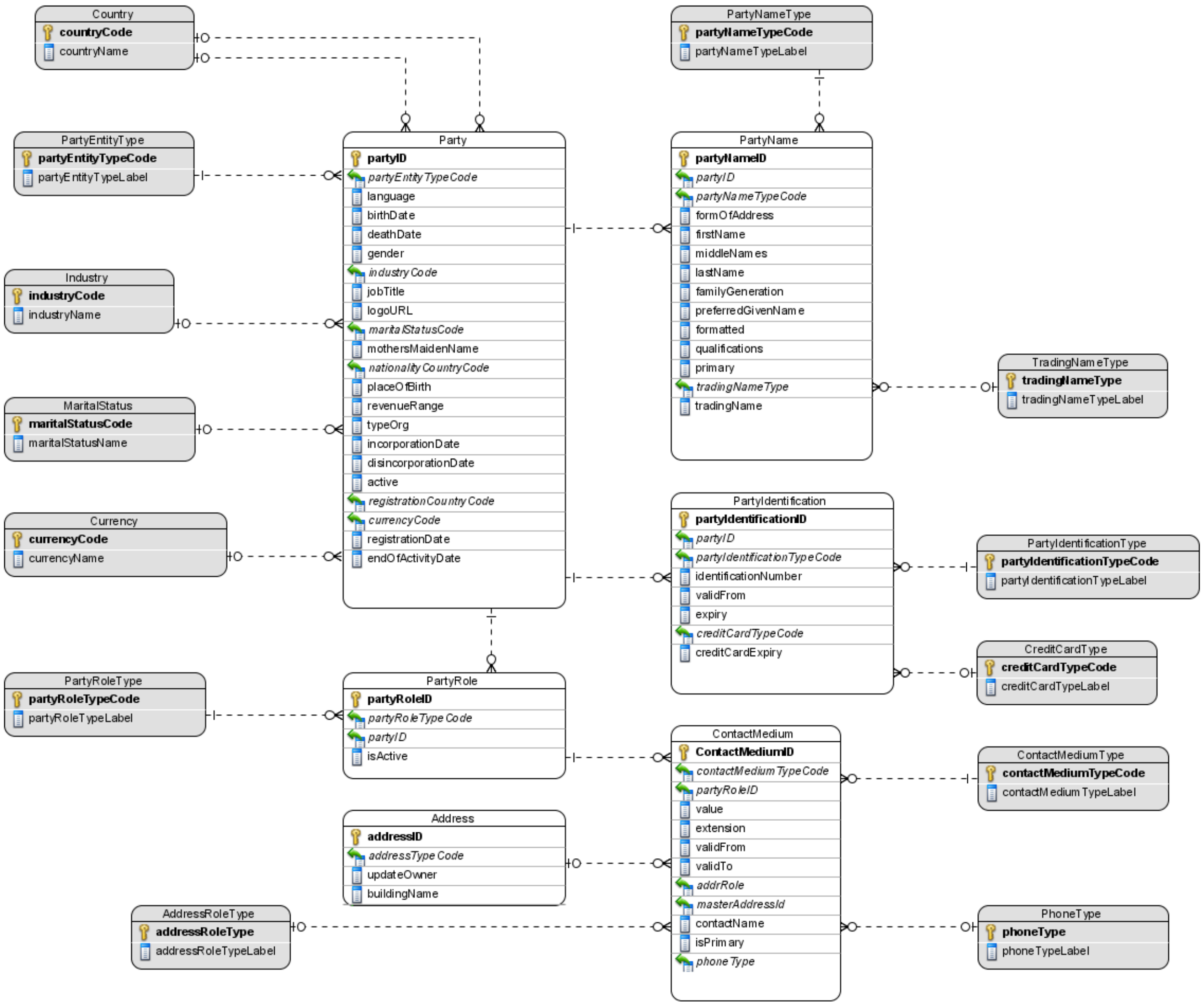
8.1.11 CountryValidationRules (countryValidationRulesDoc) / CWT_COUNTRYVALIDATIONRULES

A cross reference table of address field names and country codes. This table is used to provide validation rules on address fields by country and is populated using the Configuration menu.

Attribute/Column Name	Type	Description
(PFK)fieldNameCode	string(16)	FK to field name.
(PFK)countryCode	string(16)	FK to country code.
hidden	boolean	Field not displayed if true.
mandatory	boolean	Field required if true.
rule	string(64)	Regular expression to be executed to provide data validation. For regular expressions syntax refer to http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:Regular_Expressions

Attribute/Column Name	Type	Description
errorMessage	string(64)	Error message in the event validation fails.

8.2 Parties Entity



8.2.1 PartyEntityType

Code table. Values are Individual or Organization.

Attribute/Column Name	Type	Description
(PK)partyEntityTypeCode	string(1)	Unique identifier. ‘I’ or ‘O’.
partyEntityTypeLabel	string(64)	

8.2.2 Industry

Code table of industry types.

Attribute/Column Name	Type	Description
(PK)industryCode	string(16)	Unique identifier.
industryName	string(64)	

8.2.3 MaritalStatus

Code table of marital status values including:

- SI – Single
- MA – Married
- CL – Common Law
- WI – Widowed

Attribute/Column Name	Type	Description
(PK)maritalStatusCode	string(2)	Unique identifier.
maritalStatusName	string(64)	

8.2.4 Currency

Code table of currency codes such as USD, CAD, Euro, etc.

Attribute/Column Name	Type	Description
(PK)currencyCode	string(16)	Unique identifier.
currencyName	string(64)	

8.2.5 PartyNameType

Code table of party name types. Values include ‘Short Name’, ‘Operating Name’, etc.

Attribute/Column Name	Type	Description
(PK)partyNameTypeCode	string(16)	Unique identifier.
partyNameTypeLabel	string(64)	

8.2.6 TradingNameType

Code table of types applied to a trading (organization) name. Values include ‘Co.’, ‘Inc.’, ‘Partnership’, etc.

Attribute/Column Name	Type	Description
(PK)fieldNameCode	string(16)	Unique identifier.
fieldName	string(64)	

8.2.7 PartyIdentificationType

Code table of identification types. Values include ‘Birth Certificate’, ‘Driver’s License’, ‘Credit Card’, etc.

Attribute/Column Name	Type	Description
(PK)partyIdentTypeCode	string(16)	Unique identifier.
partyIdentTypeLabel	string(64)	

8.2.8 CreditCardType

Code table of valid credit card types. Values include ‘VISA’, ‘Mastercard’, ‘AMEX’, etc.

Attribute/Column Name	Type	Description
(PK)fieldNameCode	string(16)	Unique identifier.
fieldName	string(64)	

8.2.9 Party

A party is a person or an organization. A party may be known by one or more names and have multiple forms of identification such as birth certificate, drivers licence, etc.

A party may assume multiple roles. The role may be a customer, account manager, sales representative, customer contact, etc.

This is a simplification of the SID model. The individual and Organization entities were combined.

Attribute/Column Name	Type	Description
(PK)partyId	string(16)	Unique identifier.
(FK)partyEntityTypeCode	string(1)	Type of Party: ‘I’ for Individual or ‘O’ for Organization
(FK)language	string(2)	‘iso6391’ Code Table.
birthdate	date	Date of birth
deathDate	date	Date deceased.
gender	string(1)	M or F
(FK)industryCode	string(16)	Classifies party by industry.
jobTitle	string(50)	Job title
logoURL	string(128)	Organization URL

Attribute/Column Name	Type	Description
(FK)maritalStatusCode	string(2)	
nationality	string(2)	iso3166 code table
mothersMaidenName	string(35)	Individual’s Mother’s Maiden Name
placeOfBirth	string(30)	Individual’s place of birth
revenueRange	string(16)	6 codes, ranging from less than 100,000 (code 0) to more than 100,000,000 (code 5)
typeOrg	string(10)	Organization type. Free format.
incorporationDate	date	Date of Incorporation
disincorporationDate	date	Date corporation dissolved.
active	number(1)	
registrationCountryCode	string(2)	Country of residence
currencyCode	string(2)	Preferred currency
registrationDate	date	
endOfActivityDate	date	
inorganization	Number(1)	Specifies if Customer Contact relates to any Organization (in which case Job Title is defined). Used only for Organizations.

8.2.10 PartyName

A party may be known by multiple names. This entity records the names of the party. This is a simplification of the SID concept.

Attribute/Column Name	Type	Description
(PK)partyNameID	string(16)	Unique identifier.
(FK)partyId	string(16)	
partyNameTypeCode	string(16)	A code which indicates the ‘type’ of name. For example, ‘Operating Name’, ‘Short Name’, ‘Nick Name’. Different code tables exist depending on the type of party. cwt_ContactNameType_Ind for Individuals and cwt_ContactNameType_Org for Organizations.
formOfAddress	string(10)	Mr., Mrs, etc.
firstName	string(25)	
middleNames	string(32)	
lastName	string(35)	
familyGeneration	string(1)	Jr, Sr.
preferredGivenName	string(64)	
formatted	string(128)	

Attribute/Column Name	Type	Description
qualifications	string(32)	Contains the letters used to describe academic or other type qualifications held by a person and/or the distinctions conferred upon them. For example, PhD, MD, CPA, MCSD, etc
primary	number(1)	Primary name for the party. This is the default or visual key for the party.
tradingNameType	string(16)	Code Table: cwt_OrgNameType. Examples: Co., Inc., Ltd., Pty Ltd., Plc., Gmbh
tradingName	string(50)	Trading or operating name.

8.2.11 PartyIdentification

A party may have multiple forms of identification such as a birth certificate, drivers licence, credit card, etc. This entity records these forms of identification of the party.

Attribute/Column Name	Type	Description
(PK)partyIdentificationID	string(16)	Unique identifier.
(FK)partyId	string(16)	
partyIdentTypeCode	string(16)	A code which indicates the kind of identification. Birth Certificate etc. Depending upon whether the party is an individual or Organization, different code tables are referenced. cwt_IndIdentificationType and cwt_OrgIdentificationType.
identificationNumber	string(32)	The identification number.
validFrom	date	From date.
expiry	date	To date.
creditCardType	string(16)	Credit card type… VISA, MC, etc. Code Table: cwt_creditCardType
creditCardExpiry	string(6)	YYYYMM

8.2.12 PartyRole

A party may assume multiple roles. For example a party may have the role of customer and primary contact. This entity records the roles a party ‘plays’. Generally, subtypes of this entity exist to provide role specific attributes.

Attribute/Column Name	Type	Description
(PK)partyRoleID	string(16)	Unique identifier.
(FK)partyId	string(16)	
(FK)partyRoleTypeCode	string(16)	Code Table: cwt_party_PartyRole
isActive	number(1)	

8.2.13 ContactMedium

Used to record contact means for parties by party role. For example, this entity may record the email address for a customer (a party who has the role of customer). If the contact medium is a mailing address, then this entity keeps a reference to the address table.

The number/address field can be validated accordingly to regular expression provided by the cwt_contactTypeValidation code table. The system will look for the same contact medium code into this table and, if found, will verify the content of the number/address field using the regular expression provided in the description column.

If a new document needs to reuse the contact medium functionality provided by the module, it must:

- Be an extension of the Party Role Document or, if the Contact Medium Finder is being invoked, the search document must have a field name "refPartyRole" carrying the party role id.

- Provide a conversion map from the document to the ‘Search - Contact Medium’ document. The conversion map must provide the refParty (id of the party) and the refPartyRole (id of the party role). This will allow the finders to retrieve the contact mediums related to the current party role and the contact mediums related to all roles played by the party.

- Implement a button to invoke the Contact Medium Finder through “cwt_party.popContactMediums” script or add a Reference field to Contact Medium, if the document needs a 1:1 relation to the Contact Medium.

Attribute/Column Name	Type	Description
(PK)contactMediumID	string(16)	Unique identifier.
(FK)contactMediumTypeCode	string(16)	Code Table: cwt_contactMediumType
(FK)partyRoleID	string(16)	
value	string(64)	Text of Contact medium. For example, if the type is an email address, this field contains the actual address.

Attribute/Column Name	Type	Description
extension	number(6)	Phone extension
validFrom	date	
validTo	date	
(FK)addrRole	string(16)	Code Table: cwt_addressRole When entering a Postal Address contact medium, the user may specify a role for the address, like 'Business Address'
(FK)masterAddressID	string(32)	If a Postal Address contact medium is being entered, a Master Address Reference field will be used to point to the address information. The concept of Master Address, where all addresses can be reused, is active only if the 'useMasterAddress' global configuration variable is set to 'true'. Otherwise, each postal address contact medium will point to a unique address document that cannot be reused.
contactName	string(64)	Name of contact.
isPrimary	number(1)	
phoneType	string(1)	Enumeration: H – Home W – Work M – Mobile O - Other

8.2.14 ContactMediumType / cwt_contactMediumType

Code table of valid contact mediums. Values include Email, Fax, Postal Address, etc. Some medium types are declared as constants in metadata and cannot be changed or deleted. This includes, 'POSTALADDR', 'PHONE', 'FAX'.

Attribute/Column Name	Type	Description
(PK)contactMediumTypeCode	string(16)	Unique identifier.
contactMediumTypeLabel	string(64)	

8.2.15 AddressRoleType

Code table of valid address roles. Values include Home, Business, etc.

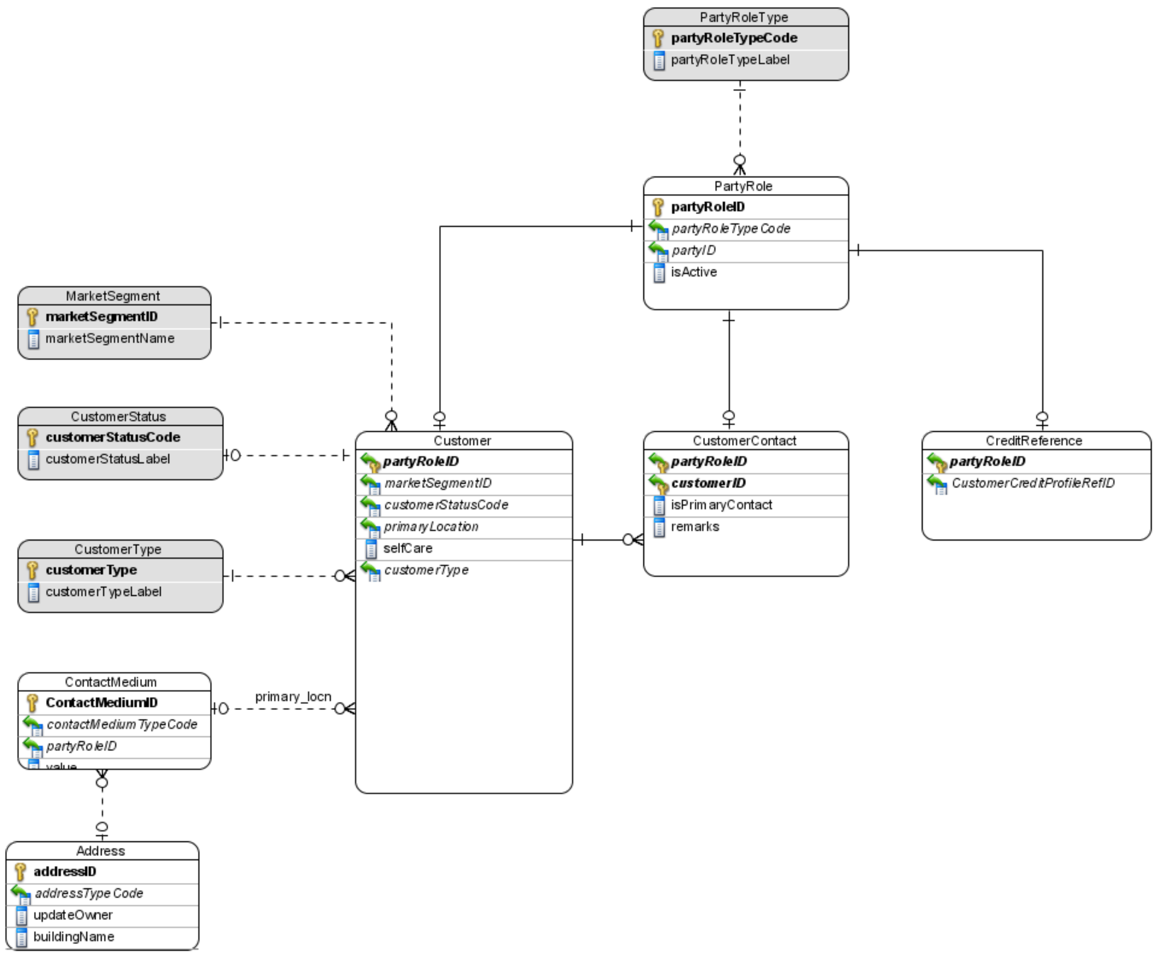
Attribute/Column Name	Type	Description
(PK)addressRoleCode	string(16)	Unique identifier.
addressRoleLabel	string(64)	

8.2.16 PhoneType

Code table of valid phone types. Values include Home, Mobile, Work etc.

Attribute/Column Name	Type	Description
(PK)phoneTypeCode	string(1)	Unique identifier.
phoneTypeLabel	string(64)	

8.3 Party Roles Entity



8.3.1 MarketSegment

Code table of valid market segments.

Attribute/Column Name	Type	Description
(PK)marketSegmentID	string(16)	Unique identifier.
marketSegmentName	string(64)	

8.3.2 CustomerStatus

Code table of valid customer status codes.

Attribute/Column Name	Type	Description
(PK)customerStatusCode	string(16)	Unique identifier.
customerStatusLabel	string(64)	

8.3.3 CustomerType

Code table of valid customer type codes.

Attribute/Column Name	Type	Description
(PK)customerType	string(16)	Unique identifier.
customerTypeLabel	string(64)	

8.3.4 PartyRoleType

Code table of valid party role types. Values include ‘CUSTOMER’, ‘CUSTOMERCONTACT’, etc. This code table is particularly important as it defines the list of valid subtypes of PartyRole. Invariably, constants are defined is metadata representing these sub-types.

Attribute/Column Name	Type	Description
(PK)partyRoleTypeCode	string(16)	Unique identifier.
partyRoleTypeLabel	string(64)	

8.3.5 Customer

A sub-type of PartyRole. A customer is a party assuming the role of customer. This table/entity contains only those attributes and relationships which are specific to customers.

Attribute/Column Name	Type	Description
(PFK)partyRoleID	string(16)	Unique identifier.
(FK)marketSegmentID	string(16)	Market Segment
(FK)customerStatusCode	string(12)	Current customer status
(FK)primaryLocation	string(16)	FK to address table identifying primary address
selfCare	number(1)	If true, customer has self care.
(FK)customerType	string(3)	Code Table: cwt_customerType
customerSubType	VARCHAR2(3)	Customer Subtype
serviceproviderid	VARCHAR2(16)	Service provider
isserviceprovider	number(1)	Is customer a service provider

8.3.6 CustomerContact

A sub-type of PartyRole. A customer contact is a party (individual) assuming the role of 'customer contact'. This table/entity contains only those attributes and relationships which are specific to customer contacts.

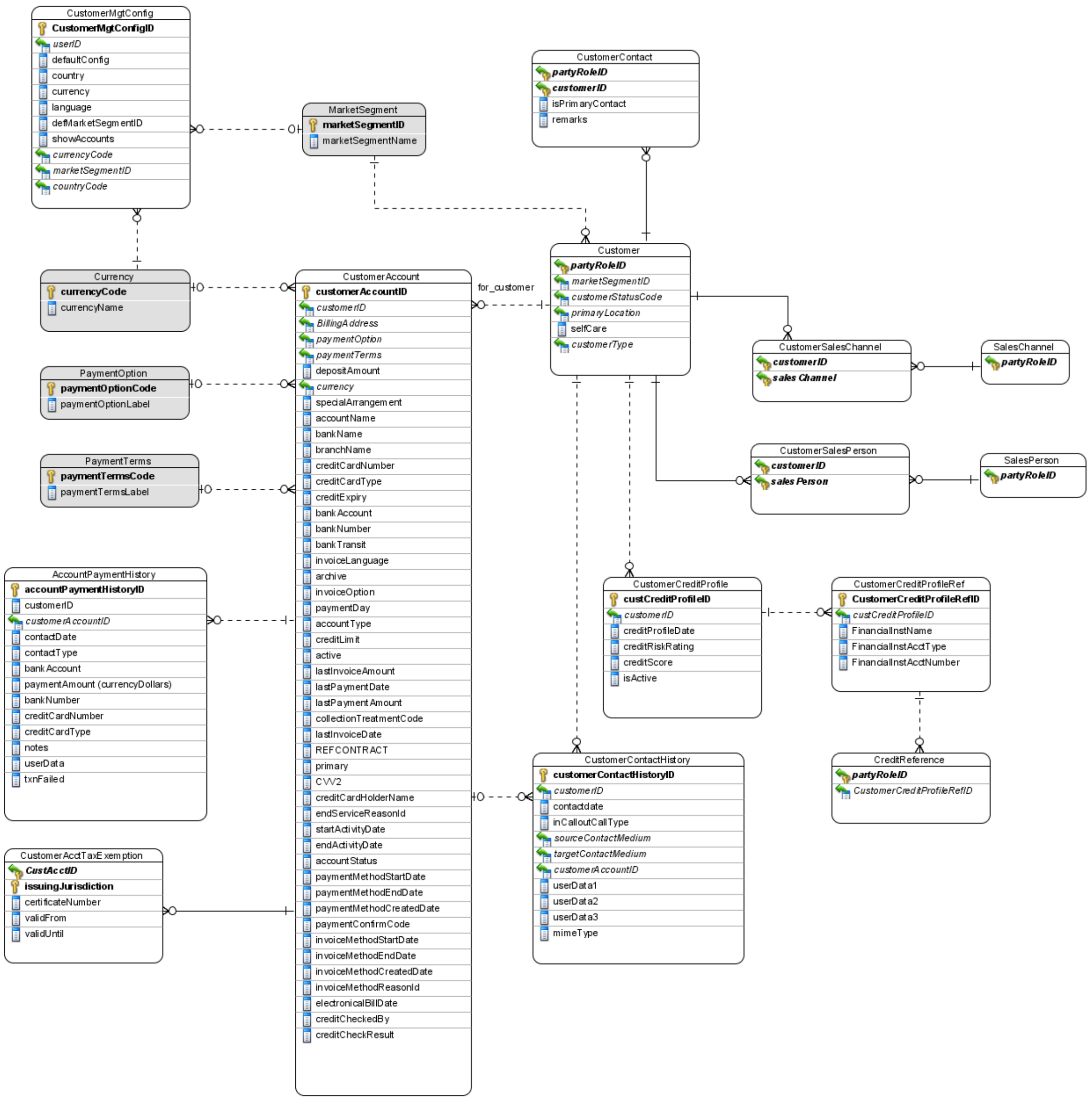
Attribute/Column Name	Type	Description
(PFK)partyRoleID	string(16)	Unique identifier.
(FK)customerID	string(16)	FK to customer.
primary	number(1)	If true, is primary contact for the customer.
remarks	string(250)	Free format remarks.
refcustomersite	VARCHAR2(16)	Relates a customer to a site.
refcustomeraccount	VARCHAR(16)	Relates a customer to an account.
siteresponsibility	VARCHAR(16)	
(PFK)partyRoleID	string(16)	Unique identifier.
(FK)customerID	string(16)	FK to customer.

8.3.7 CreditReference

A sub-type of PartyRole. A credit reference is an individual or organization who assumes the role of Credit Reference. This table/entity contains only those attributes and relationships which are specific to a credit reference.

Attribute/Column Name	Type	Description
(PFK)partyRoleID	string(16)	FK to Party Role.
(FK)customerCreditProfileRefID	string(16)	FK to credit reference details.

8.4 Customers Entity



8.4.1 PaymentOption

Code table of valid payment options.

Attribute/Column Name	Type	Description
(PK)paymentOptionCode	string(16)	Unique identifier.
paymentOptionLabel	string(64)	

8.4.2 PaymentTerms

Code table of valid payment terms.

Attribute/Column Name	Type	Description
(PK) paymentTermsCode	string(16)	Unique identifier.
paymentTermsLabel	string(64)	

8.4.3 CustomerMgtConfig

System configuration table by user. Options within this table affect system behavior for application users.

Attribute/Column Name	Type	Description
(PK) customerMgtConfigID	string(16)	Unique identifier.
(FK)userID	string(64)	FK to user defined using User Profile Management.
defaultConfig	number(1)	
(FK)country	string(2)	
(FK)currency	string(3)	
(FK)language	string(2)	
(FK)defMarketSegmentID	string(16)	
showAccounts	number(1)	

8.4.4 CustomerAccount

A customer will typically have only one (billing) account but could have more than one. The customer account is used for billing purposes.

Attribute/Column Name	Type	Description
(PK)customerAccountId	string(16)	Unique identifier.
(FK)customerId	string(16)	FK to customer
(FK)billingAddressID	string(16)	FK to contact medium where medium is an address.
(FK)paymentOption	string(6)	Enumeration: Credit, Debit, Bill
paymentTerms	string(3)	Enumeration: Net 30, Net 60, On Receipt, Special Arrangement.
depositAmount	number(10,2)	
(FK)currency	string(8)	Code Table: cwt_billingCurrency
specialArrangement	string(250)	
accountName	string(64)	
bankName	string(30)	
branchName	string(30)	
creditCardNumber	string(16)	
(FK)creditCardType	string(16)	Code Table: cwt_creditCardType

Attribute/Column Name	Type	Description
creditExpiry	string(6)	
bankAccount	string(32)	
bankNumber	string(3)	
bankTransit	string(5)	
(FK)invoiceLanguage	string(8)	Code Table: cwt_invoiceLanguage
archive	number(1)	
(FK)invoiceOption	string(10)	Code Table: cwt_dictInvoiceOptions
paymentDay	string(2)	Enumeration: 1-28
accountType	string(8)	
creditLimit	number(6)	
active	number(1)	
lastInvoiceAmount	number(10,2)	
lastPaymentDate	date	
lastPaymentAmount	number(10,2)	
collectionTreatmentCode	string(32)	
lastInvoicedDate	date	
primary	number(1)	
CVV2	string(6)	
creditCardHolderName	string(32)	
endServiceReasonID	string(12)	
startActivityDate	date	
endActivityDate	date	
(FK)accountStatus	string(12)	Code Table: accountStatusCT
paymentMethodStartDate	date	
paymentMethodEndDate	date	
paymentMethodCreatedDate	date	
invoiceMethodStartDate	date	
invoiceMethodEndDate	date	
invoiceMethodCreatedDate	date	
paymentConfirmationCode	string(12)	
electronicBillDate	date	
creditCheckedBy	string(64)	
(FK)creditCheckResult	string(3)	Code Table: cwt_creditCheckResult
refcustomeraccount	Varchar(64)	Reference to the parent account

8.4.5 CustomerAccount

A customer will typically have only one (billing) account but can have more than one. The customer account is used for billing purposes.

Attribute/Column Name	Type	Description
(PK)customerAccountId	string(16)	Unique identifier.

Attribute/Column Name	**Type**	**Description**
(FK)customerId	string(16)	FK to customer
(FK)billingAddressID	string(16)	FK to contact medium where medium is an address.
(FK)paymentOption	string(6)	Enumeration: Credit, Debit, Bill
paymentTerms	string(3)	Enumeration: Net 30, Net 60, On Receipt, Special Arrangment.
depositAmount	number(10, 2)	
(FK)currency	string(8)	Code Table: cwt_billingCurrency
specialArrangement	string(250)	
accountName	string(64)	
bankName	string(30)	
branchName	string(30)	
creditCardNumber	string(16)	
(FK)creditCardType	string(16)	Code Table: cwt_creditCardType
creditExpiry	string(6)	
bankAccount	string(32)	
bankNumber	string(3)	
bankTransit	string(5)	
(FK)invoiceLanguage	string(8)	Code Table: cwt_invoiceLanguage
archive	number(1)	
(FK)invoiceOption	string(10)	Code Table: cwt_dictInvoiceOptions
paymentDay	string(2)	Enumeration: 1-28
accountType	string(8)	
creditLimit	number(6)	
active	number(1)	
lastInvoiceAmount	number(10, 2)	
lastPaymentDate	date	
lastPaymentAmount	number(10, 2)	
collectionTreatmentCode	string(32)	
lastInvoicedDate	date	
primary	number(1)	
CVV2	string(6)	
creditCardHolderName	string(32)	
endServiceReasonID	string(12)	
startActivityDate	date	
endActivityDate	date	
(FK)accountStatus	string(12)	Code Table: accountStatusCT
paymentMethodStartDate	date	

Attribute/Column Name	Type	Description
paymentMethodEndDate	date	
paymentMethodCreatedDate	date	
invoiceMethodStartDate	date	
invoiceMethodEndDate	date	
invoiceMethodCreatedDate	date	
paymentConfirmationCode	string(12)	
electronicBillDate	date	
creditCheckedBy	string(64)	
(FK)creditCheckResult	string(3)	Code Table: cwt_creditCheckResult
refcustomeraccount	Varchar(64)	Reference to the parent account

8.4.6 AccountPaymentHistory

Represents a payment made on the account.

Attribute/Column Name	Type	Description
(PK) accountPaymentHistoryId	string(16)	Unique identifier.
(FK)customerAccountID	string(16)	
(FK)customerID	string(16)	
contactDate	date	
contactType	string(1)	InBound/OutBound
bankAccount	string(32)	
paymentAmount	number(10, 2)	Actual field name is currencyDollars.
bankNumber	string(3)	
creditCardNumber	string(16)	
(FK)creditCardType	string(16)	Code Table: cwt_creditCardType
notes	string(256)	
userData	string(64)	
txnFailed	number(1)	

8.4.7 CustomerAcctTaxExemption

Indicates whether a customer is exempt from paying a particular tax.

Attribute/Column Name	Type	Description
(PFK)customerAccountID	string(16)	Unique identifier.
issuingJurisdiction		
certificateNumber		
validFrom		
validUntil		

8.4.8 CustomerContactHistory

Tracks communication with a customer.

Attribute/Column Name	Type	Description
(PK)customerContactHistoryID	string(16)	Unique identifier.
(FK)customerID	string(16)	
(FK)customerAccountID	string(16)	
contactDate	date	
type	string(1)	InBound/OutBound
(FK)sourceContactMedium	string(16)	FK to source CM
(FK)targetContactMedium	string(16)	FK to target CM
mimeType	string(32)	
userData1	string(64)	
userData2	string(64)	
userData3	string(64)	

8.4.9 CustomerCreditProfile

Attribute/Column Name	Type	Description
(PK)customerCreditProfileID	string(16)	Unique identifier.
(FK)customerID	string(16)	
creditProfileDate	date	
creditRiskRating	string(1)	Enumeration: 1 star, 2 stars, etc.
creditScore		
active	number(1)	

8.4.10 CustomerCreditProfileRef

Attribute/Column Name	Type	Description
(PK)customerCreditProfileRefID	string(16)	Unique identifier.
(FK)customerCreditProfileID		
financialInstName	string(32)	
financialInstAcctType	string(32)	
financialInstAcctNumber	string(32)	

9 Configuration Variables

The table below documents all configuration variables used in the modules.

Variable Name	Values	Default Value	Action
logLevel	0 to 7	3	Defines the log level of the modules. 0 - emergency logs only 7 - all logs, including Debug, such as syslog standard log levels.
useMasterAddress	true / false	false	Enables / disables Master Address
googleMapUrl		http://maps.google.com/maps/api/staticmap?center=	Google maps API url. Used to show addresses as clickable links.
googleUrlParams		&zoom=14&size=400x400&sensor=true	Google maps zoom level and panel size.

10 Error Codes

Error Code	Definition
TECUST001	Document not found for update
TECUST002	Document not found for update
TECUST003	Document not found for update
TECUST004	Could not create Party of the Account Contact
TECUST005	No search criteria provided
TECUST006	Missing mandatory contact mediums for party role
TECUST007	Please select only one row to be set as primary
TECUST008	Invalid address field - validation rules provided by user
TECUST009	Party is not Unique
TECUST010	Cannot delete primary party name
TECUST011	Cannot delete mandatory contact medium if it is the only one left
TECUST012	Invalid / Incomplete Party
TECUST013	Customer document is invalid
TECUST014	no id provided
TECUST015	id not found in DB
TECUST016	invalid parameter
TECUST017	Please select one item
TECUST018	Cannot delete a party that is playing roles
TECUST019	Cannot delete a primary party name
TECUST020	Returned by the Search API in the response structure,

Error Code	Definition
	if the data returned is incomplete (e.g. a search for customers could return with missing contact mediums). If this error happens, you should narrow the search criteria.
TECUST021	Returned by the Search API in the response structure, if more rows than the maximum specified were found for the given search criteria.

11 Customer Module API

Data entity objects in the Customer module provide search, create and delete functionalities. CIM module uses that functionality to build Customer 360 Tree and Site 360 Tree.

11.1 Customer Document

A customer document has the following methods:

createAccount(), createContact(), createExternalIdentifier(), createNote(), createSite(),

deleteAccounts(acctIds), deleteContacts(contactIds), deleteExternalIdentifiers(extIds),

deleteNotes(noteIds), deleteSites(siteIds), getAccounts(), getAllContacts(), getAllExternalIdentifiers(), getAllNotes(), getContacts(), getExternalIdentifiers(), getNotes(), getSites()

11.2 Customer Contact

Customer Contact is an extension of Party Role with a fixed roleType defined by the “cwt_party.partyRoleCustomerContact()” script. As an extension of Party Role, it can be played by any Party in the database.

Besides the inherited party role fields, a Customer Contact Document has:

- A reference to Customer: one Customer can have many Customer Contacts, but a Contact Document will always be owed by a Customer. If the references to an Account and a Site are not null the contact is considered to be a Customer contact.

- A reference to Account: one Account can have many Account Contacts, but a Contact Document can be owed by only one Account. If the reference to an Account is not null the contact is considered to be an Account Contact.

- A reference to Site: one Site can have many Site contacts, but a Contact Document can be owned by only one Site. If the reference to a Site is not null the contact is considered to be a Site Contact.
- Remarks: this is a free text field.
- Primary flag: indicates this is the primary contact for the Customer. There can be only one primary contact per Customer. If a new Customer Contact is defined primary, the old primary contact is marked as non-primary before saving the new one.

In the UI, when creating a new Customer Contact, all fields will be read-only until the Party is defined. Contact (Customer Contact, Account Contact or Site Contact) has always the type “Individual”.

Customer Contact Document inherits the following methods from Party Role Document:

createContactMedium(), createParty(), deleteContactMediums(), deleteParty(), destroy(), getAllContactMediums(), getContactMediumById(id), getParty()

11.3 Customer Account Document

Customer Account Document includes the following methods:

createAccountInvoice(),createBillingAddressCM(),createContact(),

createContactMedium(),createExternalIdentifier(),createInvoice(),createNote(),

createPaymentMethod(),deleteAccountInvoices(invIds),

deleteBillingAddressCM(),

deleteContacts(contactIds), deleteExternalIdentifiers(extIds),

deleteNotes(noteIds),

deletePaymentMethods(pmIds),

deleteSites(),getAllAccountInvoices(),getAllContactMediums(),

getAllPaymentMethods(),getAllPrimaryExternalIdentifiers(),getBillingAddressCM(),

getContactMediumById(),getContacts(),getExternalIdentifiers(),getNotes(),

getPrimaryContactMedium(),getSites()

11.4 Customer Site Document

The main purpose of the document is to keep service address information along with other site specific information.

Customer document supports the following methods.

createContact(),createContactMedium(),createExternalIdentifier(), createNote(),

deleteContacts(contactIds), deleteExternalIdentifiers(extIds), deleteNotes(noteIds),

getAllContactMediums(),getAllPrimaryExternalIdentifiers(),getContactMediumById(),

getContacts(),getExternalIdentifiers(),getNotes(),getPrimaryContactMedium(),

getPrimaryContactMedium()

11.5 Party Document

Party document has the following methods:

createIdentification(),createPartyName(),deleteIdentifications(pieceIds), deletePartyNames(nameIds), getIdentifications(),getPartyNames()

11.6 Party Role Document

Party Role document has the following methods:

createContactMedium(),createParty(),deleteContactMediums(),deleteParty(), deletePartyOK(),getAllContactMediums(),getContactMediumById(id), getParty(),getPrimaryContactMedium(type)

11.7 Contact Medium Document

Contact Medium document has the following methods:

getAddress(),getPartiRole()

12 CIM Extension Points

12.1 Global Script Functions

The name convention of the functions to be overwritten starts with '_' (underscore). The CIM module only provides the default implementation for the following functions:

- function cwt_cim._getDefaultCurrency()

  This function returns the default currency for the payment unit, the default CIM implementation returns 'CAD' for Canadian dollar.

- function cwt_cim. _newQuote(customerId, accountId)

  customerId – string input parameter, customer ID;

  accountId – string input parameter, account ID;

  This function opens the quote/order page allowing the user to create a new quote/order. The default CIM implementation only prompts a message indicating the customer logic should be implemented.

- function cwt_cim. _openQuote(orderId)

  orderId – string input parameter, order ID;

  This function opens the quote/order page allowing the user to view or modify an existing quote/order. The default CIM implementation only prompts a message indicating the customer logic should be implemented.

- function cwt_cim._quoteFinderSelect(customerId, accountId)

  customerId – string input parameter, customer ID;

  accountId – string input parameter, account ID;

  This function returns an array of cwt_cim. qoInformationDetailDoc document objects, it allow to display the list of quote/order under the customer tree, the default CIM quotes implementation is an empty method. The application must overwrite the function to list the quotes/orders related to the customer account.

12.2 Documents in the cwt_cim namespace

In principle, every document in the cwt_cim namespace can be extended/overwritten, but the most important ones affecting the business logic and display are the documents serving the nodes of the customer tree display.

- cwt_cim.**custSummaryDoc** – Customer Summary Document, on Customer;
- cwt_cim.**contactDoc** – Contact document for Customer/Account/Site
- cwt_cim.**accountDoc** – Account Document
- cwt_cim.**invoiceDoc** – Invoice Document
- cwt_cim.**paymentMethodDoc** – Payment Method Document
- cwt_cim.**srResultDoc** – Service Registry Result Document, used as the output of Account Service Registry Finder;
- cwt_cim.**siteGeneralInfo** – Document used to keep site general information.
- cwt_cim.**qoInformationDetailDoc** - Quote/Order Information – Detail document
- cwt_cim.**contactMediumNodeDoc** – Contact Medium document
- cwt_cim.**partyIdentificationDoc** – Party Identification Document, used as the output of Party Identification Finder;
- cwt_cim.**extIdentifierDoc** – Customer/Account/Site external identifiers
- Among the methods associated with the document, the most important ones are **doSave()** and **doDelete()**, which should be verified to see if the default logic satisfies the application business logic when doing save and delete, if not, the logic are to be overwritten.

12.3 Finders in the cwt_cim namespace

In principle, every finder in the cwt_cim namespace can be overwritten, but the most important ones affecting the business logic and display are the finders serving the nodes of the customer tree display.

- cwt_cim.**contactFinder** – Customer/Account/Site contacts finder
- cwt_cim.**noteFinder** – Notes finder
- cwt_cim.**cusTreeAccountFinder** – Customer Accounts finder
- cwt_cim.**invoiceFinder** – Invoice Finder
- cwt_cim.**paymentMethodFinder** – Payment Method Finder
- cwt_cim.**accountSRFnd** – Account Service Registry Finder

- cwt_cim.**cusTreeSiteFinder** – Customer Tree Site Finder- searches for a site based on a given service address
- cwt_cim.**quoteOrderFnd** – Quote/Order Finder, under Customer;
- cwt_cim.**historyFinder** – History finder
- cwt_cim.**contactMediumNodeFnd** – contact Medium Finder
- cwt_cim.**refExtIdentiferFnd** – External Identifier Finder
- cwt_cim.**partyIdentificationFnd** – Party Identification Finder

Among the methods associated with the finders, the most important ones are **cwOnFinderSel ()** for the query logic, and **doubleClick()** for the behavior. When double-clicking the highlighted item, the default logic verifies if the application business logic is satisfied. If not, the logic is overwritten.

The UI display for the extension points are:

- The tree node under Quotes/Orders, and the list results of Quote/Order Finder are from the ON module;
- The finder results list of the Service Registry tab on Customer/Accounts is from the ON module.

13 CIM API

The CIM module provides JavaScript API to be used inside the Velocity Studio AVM metadata, and web service API to be used by external system.

13.1 JavaScript API

The API is exposed through Customer and Notification APIs

13.1.1 Customer APIs

The class CustomerDataObject, which is implemented using metadata DataStructure, exposes the member functions:

- function cwt_cim.getCustomerDataObject(customerId)

  customerId – string input parameter, customer ID;

  This function returns the **CustomerDataObject** associated with the given customer id.

- CustomerDataObject.getCustomerDetailInfo()

  This function returns the customer detail info object (for individual or organization customer).

- CustomerDataObject.getAccountInfo(accountId)

  accountId – string input parameter, account ID;

  This function returns the account info object of the given account id.

- CustomerDataObject.updateAccountInfo(accountId, info)

  accountId – string input parameter, account ID;

  info – object input parameter, new account info;

  This function updates the account with new info.

- CustomerDataObject.getAccountDS(acctId)

  acctId – string input parameter, account ID;

  This function return a cwt_cust:customerAccount structure related to the given acctId.

- CustomerDataObject.getCustomerDS()

  This function returns the cwt_cust:customerAccount associated with the current customer.

- CustomerDataObject.getExternalID(ownerId, externalSystem, id)

  ownerId – string input parameter;

  externalSystem – string input parameter;

  id – string input parameter;

  This function returns an array of external system id objects.

- CustomerDataObject.saveExternalID(extId)

  extId – object input parameter;

  This function saves the external system id object.

- CustomerDataObject.getAddressDetail(addressId, ownerId, addressType, isPrimary)

  addressId – string input parameter;

  ownerId – string input parameter;

  addressType – string input parameter;

isPrimary – boolean input parameter;

This function returns the address detail object.

- CustomerDataObject.getCMSById(cmId, ownerId, cmType, hasRelated)

  cmId – string input parameter;

  ownerId – string input parameter;

  cmType – string input parameter;

  hasRelated – boolean input parameter;

  This function returns the contact medium object by id or an array of contact medium objects if cmId is null.

  CustomerDataObject.getServiceAddressDetail(siteId)

  siteId – site id for which a service address is being requested

  This function returns a service address for a given site id.

13.1.2 Notification APIs

The Notification APIs expose the following functions (for more details on the APIs refer to the Java documents in the product documentation (<product installation folder> \modules\documentation\api_javadoc\index.html):

- createNotificationTemplateByValue(notificationTemplate template): Creates a notification template with internationalizable information.

- getNotificationTemplateByKey(notificationTemplate template): Retrieves information about a notification template.

- queryNotificationTemplate(notificationTemplateSearch search): Searches for a list of notification templates matching the criterion.

- removeNotificationTemplateByKey(notificationTemplate template): Removes a notification template from system.

- sendEmailMessage(message message, java.lang.String statusHandler, java.lang.String contentHandler)

- sendNotificationMessage(message message)

- sendSMSMessage(message message, java.lang.String statusHandler, java.lang.String msgHandler)

- updateMesageStatus(receipt input, java.lang.String statusHandler)

- updateNotificationTemplateByValue(notificationTemplate template): Updates an existing notification template, complete list of internationalizable content must be provided.

13.2 Web Service API

The web service APIs are implemented in a separate namespace and exposes basic IM functionality.

Check the response metadata type to see if the Web Service call returned any errors. If the metadata type returned was *CIMFault*, the Web Service call has failed. Otherwise, a corresponding output datastructure is returned.

getCustomerByPhone – provides customer id by customer phone

Input: CIMGetCustomerByPhoneRequest datastructure

phone

Output: CIMGetCustomerByPhoneResponse

Customerid

getCustomerByKey – provides customer data by customer id

Input: CIMGetCustomerByKeyRequest

Customerid

Output: CIMGetCustomerByKeyResponse

getCustomerByEmail - provides customer id by customer email

Input: CIMGetCustomerByEmailRequest

Email

Output: CIMGetCustomerByEmailResponse

getCustomersByValues – returns customer details by search parameters

Input**:** CIMGetCustomersByValuesRequest

Output: CIMGetCustomersByValuesRequest

getCustomerPhones – returns customer phones by customer id

Input: CIMGetCustomerPhonesRequest

Customer id

Output: CIMGetCustomerPhonesResponse

getCustomerEmails - returns customer emails by customer id

Input: CIMGetCustomerEmailsRequest

customer id

Output: CIMGetCustomerEmailsResponse

getCustomerContactMediums – returns customer's contact mediums

Input: CIMGetCustomerContactMediumsRequest

customerId

contacted

Output: CIMGetCustomerContactMediumsResponse

createCustomerByValue – returns customer's information

Input: CIMCreateCustomerByValueRequest

Output: CIMCreateCustomerByValueResponse

getCustomerPersonalIdentifications – returns customer/customer contact personal identifications by customer/customer contact id

Input: CIMGetCustomerPersonalIdentificationsRequest

customerId

contacted

Output: CIMGetCustomerPersonalIdentificationsResponse

getCustomerStatus – returns customer status

Input: CIMGetCustomerStatusRequest

customer id

Output: CIMGetCustomerStatusResponse

customerStatus

getCustomerSites – returns customer sites by customer id

Input: CIMGetCustomerSitesRequest

customer id

Output: CIMGetCustomerSitesResponse

getAccountPaymentMethodsByValue – returns account payment methods

Input: CIMGetAccountPaymentMethodsRequest

accountId

customerId

Output: CIMGetAccountPaymentMethodsResponse

getAccountInvoicesByValue – returns account invoices

Input: CIMGetAccountInvoicesbyValueRequest

Accounted

customerId

Output: CIMGetAccountInvoicesbyValueResponse

getCustomerAccountByKey

Input: CIMGetCustomerAccountByKeyRequest

accountId

customerId

Output: CIMGetCustomerAccountByKeyResponse

getCustomerSiteByKey – returns customer site by site id

Input: CIMGetCustomerSiteByKeyRequest

customerId

siteId

Output: CIMGetCustomerSiteByKeyResponse

getCustomerAccounts – returns customer accounts by customer id

Input: CIMGetCustomerAccountsByKeyRequest

customerId

Output: CIMGetCustomerAccountsByKeyResponse

createContactByValue – creates customer contact

Input: CIMCreateContactByValueRequest

Output: CIMCreateContactByValueResponse

createContactMediumsByValue – create Customer/Customer Contact contact mediums.

Input: CIMCreateContactMediumsByValueRequest

Output: CIMCreateContactMediumsByValueResponse

updateContactMediums – updates contact mediums for customer/contact id

Input: CIMUpdateContactMediumsByValueRequest

Output: CIMUpdateContactMediumsByValueResponse

getCustomerContactByValue

Input: CIMGetCustomerContactByValueRequest

first name

last name

Output: CIMGetCustomerContactByValueResponse

getPersonalIdentificationCodes – returns identification codes from the code table

Input: CIMGetPersonalIdentificationCodesRequest

identificationType

securityQuestions

Output: CIMGetPersonalIdentificationCodesResponse

updatePersonalIdentifications – updates personal identifications for customer/customer contact

Input: CIMUpdatePersonalIdentificationsRequest

Output: CIMUpdatePersonalIdentificationsResponse

createPersonalIdentifications – creates personal identifications for customer/customer contact

Input: CIMCreatePersonalIdentificationsRequest

Output: CIMCreatePersonalIdentificationsResponse

getCustomerContacts – returns customer contacts

Input: CIMGetCustomerContactsRequest

Output: CIMGetCustomerContactsResponse

getCodeTableCodes – returns codes from the table specified

Input: CIMGetCodeTableCodesRequest

Output: CIMGetCodeTableCodesResponse

updateSelfCarePassword – updates customer contact password

Input: CIMUpdateSelfCarePasswordRequest

Output: CIMUpdateSelfCarePasswordResponse

getSecurityQuestionsCodes – returns codes from SecurityQuestions table

Input: none

Output: CIMGetSecurityQuestionsCodesResponse

13.3 Migration script database model

Run the migration script to migrate the database model from one CIM version to a later version. The migration script is a part of the CIM package and is found under the SQL directory.

14 Permissions Control

Modules use privileges to control users/user group access to different forms and functions. Each module (like Order Negotiations, Order Analytics, Customer Information Management and Service Registry), contain module specific permissions definitions that enables the system administrator the ability to assign module specific privileges to user roles. It is also possible for a system administrator to create a unique privilege and assign the privilege to a role through the Administration application and the User Profile options. This section lists the pre-defined privileges for CIM.

Privilege ID	**Privilege Name**	**Functions**
cwt_cimView	CWT - Customer Read Only View	Users with this privilege cannot create orders or customers.
cwt_cimAdmin	CIM	Users with this privilege have full access to CIM (except when restricted by cwt_cimView).
cwt_addrView	CWT - Address Read Only View	Users with this privilege cannot create addresses through Address Administration.
cwtCUAddrAdmin	CWT-CU - Address Management Administrator	Users with this privilege have full access to Address Administration (except when restricted by cwt_addrView).
cwtCUAdmin	CWT-CU - Customer Management Administrator	Users with this privilege have full access to Customer Management (not intended to be used outside of CIM).
cwtCUPartyAdmin	CWT-CU - Party Management Administrator	Users with this privilege have full access to Party Management (not intended to be used outside of CIM).

15 Location Model

15.1 About This Section

This section will provide application developers with an understanding of extending the Location (Address) model.

15.2 Implementation Model / Events

The Location model is designed to capture address information used and maintained by the application.

15.2.1 Address

Address object captures information about a specific location, including the Civic address, Rural Route, Latitude / Longitude, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
LOCATION_IMP_ADDRESS_DELETE_COMPLETE	Placeholder for extended functionality after address is deleted
LOCATION_IMP_ADDRESS_LOAD	Extended mapping from the address model to the corresponding structure
LOCATION_IMP_ADDRESS_MODEL_TYPE	Determines the appropriate model based on the address structure
LOCATION_IMP_ADDRESS_MODERN_SEARCH	Returns a DataObjectList of addresses matching search criteria specified by the search key
LOCATION_IMP_ADDRESS_SEARCH	Returns a DataObjectList of addresses matching search criteria
LOCATION_IMP_ADDRESS_SEARCH_COMPLETE	
LOCATION_IMP_ADDRESS_STORE	Extended mapping from the address structure to the corresponding model
LOCATION_IMP_ADDRESS_STORE_COMPLETE	Placeholder for extended functionality after address is stored
LOCATION_IMP_ADDRESS_STRUCTURE_TYPE	Determines the appropriate structure based on the address model

15.2.2 Address Type

Address Type is an object used to capture the supported type of addresses for a country by the application. Extendibility for this object is not supported.

15.2.3 City

City is an object used to capture information about cities used and maintained by the application. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
LOCATION_IMP_CITY_MODERN_SEARCH	Returns a DataObjectList of cities matching search criteria specified by the search key
LOCATION_IMP_CITY_SEARCH	Returns a DataObjectList of cities matching search criteria
LOCATION_IMP_CITY_SEARCH_COMPLETE	

15.2.4 External Identifier

External Identifier is an object used to capture information about the identification of an address by an external system. Extendibility for this object is not supported.

15.2.5 Municipality

Municipality is an object used to capture information about municipalities that is used and maintained by the application. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
LOCATION_IMP_MUNICIPALITY_MODERN_SEARCH	Returns a DataObjectList of municipalities matching search criteria specified by the search key
LOCATION_IMP_MUNICIPALITY_SEARCH	Returns a DataObjectList of municipalities matching search criteria
LOCATION_IMP_MUNICIPALITY_SEARCH_COMPLETE	

15.3 API Event Handlers

The following are event handlers for the Location API.

15.3.1 LOCATION_ADDRESS_CREATE

Event ID	Functional Description
CW	The implementation of this event creates an address and its external identifiers. Its corresponding city or municipality, or both are created if its city code or municipality code, or both did not exist by integrating to its corresponding API event handlers: • LOCATION_CITY_CREATE • LOCATION_MUNICIPALITY_CREATE When creating an address in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_ADDRESS_MODEL_TYPE • LOCATION_IMP_ADDRESS_STORE • LOCATION_IMP_ADDRESS_STORE_COMPLETE • LOCATION_IMP_ADDRESS_STRUCTURE_TYPE • LOCATION_IMP_ADDRESS_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

15.3.2 LOCATION_ADDRESS_GET

Event ID	Functional Description
CW	The implementation of this event retrieves the detail of an address and its external identifiers. When retrieving an address in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_ADDRESS_MODEL_TYPE • LOCATION_IMP_ADDRESS_STRUCTURE_TYPE • LOCATION_IMP_ADDRESS_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

15.3.3 LOCATION_ADDRESS_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes the detail of an address. When removing an address in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_ADDRESS_MODEL_TYPE • LOCATION_IMP_ADDRESS_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.4 LOCATION_ADDRESS_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for addresses satisfying the search criterion. When searching for an address in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_ADDRESS_MODERN_SEARCH • LOCATION_IMP_ADDRESS_SEARCH • LOCATION_IMP_ADDRESS_SEARCH_COMPLETE • LOCATION_ IMP_ADDRESS_STRUCTURE_TYPE • LOCATION_ IMP_ADDRESS_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.5 LOCATION_ADDRESS_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing address and its external identifiers. Its corresponding city or municipality, or both are created if its city code or municipality code, or both did not exist by integrating to its corresponding API event handlers: • LOCATION_CITY_CREATE • LOCATION_MUNICIPALITY_CREATE When creating an address in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_ADDRESS_MODEL_TYPE • LOCATION_IMP_ADDRESS_STORE • LOCATION_IMP_ADDRESS_STORE_COMPLETE • LOCATION_IMP_ADDRESS_STRUCTURE_TYPE • LOCATION_IMP_ADDRESS_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

15.3.6 LOCATION_ADDRESS_TYPE_CREATE

Event ID	Functional Description
CW	The implementation of this event adds an address type for a country. If another address type has already been marked as the default for the country, the implementation unsets the existing default, and the new address type is marked as the default. This implementation does not provide support for extendibility when adding an address type. The implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

15.3.7 LOCATION_ADDRESS_TYPE_DELETE

Event ID	Functional Description
CW	The implementation of this event removes an address type from a country. This implementation does not provide support for extendibility when removing an address type. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.8 LOCATION_ADDRESS_TYPE_GET

Event ID	Functional Description
CW	The implementation of this event retrieves the detail of an address type for a country. This implementation does not provide support for extendibility when retrieving an address type. The implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

15.3.9 LOCATION_ADDRESS_TYPE_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for address types satisfying the search criterion. This implementation does not provide support for extendibility when searching for an address type. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.10 LOCATION_ADDRESS_TYPE_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates the detail of an address type from a country. This implementation does not provide support for extendibility when updating an address type. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

15.3.11 LOCATION_DEFAULT_ADDRESS_TYPE

15.3.12 LOCATION_CITY_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a city. This implementation does not provide support for extendibility when creating a city. The implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

15.3.13 LOCATION_CITY_DELETE

Event ID	Functional Description
CW	The implementation of this event removes a city. This implementation does not provide support for extendibility when removing a city. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.14 LOCATION_CITY_GET

Event ID	Functional Description
CW	The implementation of this event retrieves the detail of a city. This implementation does not provide support for extendibility when retrieving a city. The implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

15.3.15 LOCATION_CITY_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for cities satisfying the search criterion. When searching for a city in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_CITY_MODERN_SEARCH • LOCATION_IMP_CITY_SEARCH • LOCATION_IMP_CITY_SEARCH_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.16 LOCATION_CITY_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates the detail of a city. This implementation does not provide support for extendibility when updating a city. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

15.3.17 LOCAITON_MUNICIPALITY_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a municipality. This implementation does not provide support for extendibility when creating a municipality. The implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

15.3.18 LOCATION_MUNICIPALITY_DELETE

Event ID	Functional Description
CW	The implementation of this event removes a municipality. This implementation does not provide support for extendibility when removing a municipality. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.19 LOCATION_MUNICIPALITY_GET

Event ID	Functional Description
CW	The implementation of this event retrieves the detail of a municipality. This implementation does not provide support for extendibility when retrieving a municipality. The implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

15.3.20 LOCATION_MUNICIPALITY_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for municipalities satisfying the search criterion. When searching for a municipality in the application, the following implementation events are integrated to support extendibility: • LOCATION_IMP_MUNICIPALITY_MODERN_SEARCH • LOCATION_IMP_MUNICIPALITY_SEARCH • LOCATION_IMP_MUNICIPALITY_SEARCH_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

15.3.21 LOCATION_MUNICIPALITY_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates the detail of a municipality. This implementation does not provide support for extendibility when updating a municipality. This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

15.4 Implementation Event Handlers

15.4.1 LOCATION_IMP_ADDRESS_SEARCH

Event ID	Functional Description
CW	Looks up addresses matching all the following search criteria when provided: • Country • Address Type • Street Name (Like) • Street Type (Like) • Street number • Sub-Unit Type (Like) • Sub-Unit Number • Latitude / Longitude [Degrees / Minutes / Seconds / Direction] • PO Box Numbers • City Code • Municipality Code • Postal Code (Like) • Rural Route Identifier • Rural Route Number • Province / State

15.4.2 LOCATION_IMP_ADDRESS_MODERN_SEARCH

Event ID	Functional Description
CW	Looks up records for any entities matching any part of the search key. The following entity is considered: • ADDRESS

15.4.3 LOCATION_IMP_ADDRESS_STRUCTURE_TYPE

Event ID	Functional Description
CW	Determines the appropriate data structure to represent different addresses by address type. The following address types are represented with the Address Civic data structure: • CIVIC • POBOX • RURAL • GENERAL • OTHER

15.4.4 LOCATION_IMP_CITY_SEARCH

Event ID	Functional Description
CW	Looks up cities matching all the following search criteria when provided: • City Code • City Name (Like) • Country • Province / State • County (Like)

15.4.5 LOCATION_IMP_CITY_MODERN_SEARCH

Event ID	Functional Description
CW	Looks up records for any entities matching any part of the search key. The following entity is considered: • CITY

15.4.6 LOCATION_IMP_MUNICIPALITY_SEARCH

Event ID	Functional Description
CW	Looks up municipalities matching all the following search criteria when provided: • City Code • Municipality Code • Municipality Name (Like)

15.4.7 LOCATION_IMP_MUNICIPALITY_MODERN_SEARCH

Event ID	Functional Description
CW	Looks up records for any entities matching any part of the search key. The following entity is considered: • MUNICIPALITY

15.5 Error Codes

The following are error codes pertaining to the Location model.

15.5.1 LOC_ERR_0001

Default address type for country {0} already existed.

15.5.2 LOC_ERR_0002

Address Type was not specified when validating address information for country {0}.

15.5.3 LOC_ERR_0003

An error occurred when validating address information.

16 Customer Module

16.1 About This Section

This section provides application developers with an understanding of extending the Customer model.

16.2 Implementation Events

The customer module captures information about a customer for which businesses will typically need. The module is designed to be extendible by applications to easily include additional behavior for their specific needs. The mechanism used for extendibility is explained in detail in the Common Implementation Guide.

16.2.1 Customer

The customer object captures information about a specific customer, including the customer status and type, market segment, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_CUSTOMER_DELETE_COMPLETE	Placeholder for extended functionality after the customer is deleted
CUSTOMER_IMP_CUSTOMER_LOAD	Extended mapping from the customer model to the corresponding structure
CUSTOMER_IMP_CUSTOMER_MODEL_TYPE	Determines the appropriate model based on the customer structure
CUSTOMER_IMP_CUSTOMER_MODERN_SEARCH	Returns a DataObjectList of customers matching search criteria specified by the search key
CUSTOMER_IMP_CUSTOMER_SEARCH	Returns a DataObjectList of customers matching the search criteria
CUSTOMER_IMP_CUSTOMER_SEARCH_COMPLETE	
CUSTOMER_IMP_CUSTOMER_STORE	Extended mapping from the customer structure to the corresponding model
CUSTOMER_IMP_CUSTOMER_STORE_COMPLETE	Placeholder for extended functionality after the customer is stored
CUSTOMER_IMP_CUSTOMER_STRUCTURE_TYPE	Determines the appropriate structure based on the customer model

16.2.2 Account

The account object captures information about a specific account, including account name and status, pin, deposit, primary, and so on. For extendibility, the following events are available for applications to extend the

Event Name	Description
CUSTOMER_IMP_ACCOUNT_DEACTIVATE_STORE	
CUSTOMER_IMP_ACCOUNT_DEACTIVATE_STORE_COMPLETE	Placeholder for extended functionality after the account is deactivated
CUSTOMER_IMP_ACCOUNT_DELETE_COMPLETE	Placeholder for extended functionality after the account is deleted
CUSTOMER_IMP_ACCOUNT_LOAD	Extended mapping from the account model to the corresponding structure
CUSTOMER_IMP_ACCOUNT_MODEL_TYPE	Determines the appropriate model based on the account structure
CUSTOMER_IMP_ACCOUNT_SEARCH	Returns a DataObjectList of accounts matching the search criteria
CUSTOMER_IMP_ACCOUNT_SEARCH_COMPLETE	
CUSTOMER_IMP_ACCOUNT_STORE	Extended mapping from the account structure to the corresponding model
CUSTOMER_IMP_ACCOUNT_STORE_COMPLETE	Placeholder for extended functionality after the account is stored
CUSTOMER_IMP_ACCOUNT_STRUCTURE_TYPE	Determines the appropriate structure based on the account model

16.2.3 Contact

The contact object captures information about a specific contact, including remarks, owner customer, primary, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_CONTACT_DELETE_COMPLETE	Placeholder for extended functionality after the contact is deleted
CUSTOMER_IMP_CONTACT_LOAD	Extended mapping from the contact model to the corresponding structure
CUSTOMER_IMP_CONTACT_MODEL_TYPE	Determines the appropriate model based on the contact structure
CUSTOMER_IMP_CONTACT_SEARCH	Returns a DataObjectList of contacts matching the search criteria
CUSTOMER_IMP_CONTACT_SEARCH_COMPLETE	
CUSTOMER_IMP_CONTACT_STORE	Extended mapping from the contact structure to the corresponding model
CUSTOMER_IMP_CONTACT_STORE_COMPLETE	Placeholder for extended functionality after the contact is stored
CUSTOMER_IMP_CONTACT_STRUCTURE_TYPE	Determines the appropriate structure based on the contact model

16.2.4 Contact Medium

The contact medium object captures information about a specific contact medium, including type, value (number, address, or phone), valid from date, valid to date, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_CONTACT_MEDIUM_DELETE_COMPLETE	Placeholder for extended functionality after the contact medium is deleted
CUSTOMER_IMP_CONTACT_MEDIUM_LOAD	Extended mapping from the contact medium model to the corresponding structure
CUSTOMER_IMP_CONTACT_MEDIUM_MODEL_TYPE	Determines the appropriate model based

Event Name	Description
	on the contact medium structure
CUSTOMER_IMP_CONTACT_MEDIUM_SEARCH	Returns a DataObjectList of contact mediums matching the search criteria
CUSTOMER_IMP_CONTACT_MEDIUM_SEARCH_COMPLETE	
CUSTOMER_IMP_CONTACT_MEDIUM_STORE	Extended mapping from the contact medium structure to the corresponding model
CUSTOMER_IMP_CONTACT_MEDIUM_STORE_COMPLETE	Placeholder for extended functionality after the contact medium is stored
CUSTOMER_IMP_CONTACT_MEDIUM_STRUCTURE_TYPE	Determines the appropriate structure based on the contact medium model

16.2.5 Contact Use

The contact use object captures information about a specific contact use, including contact, contact medium, used by, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_CONTACT_USE_DELETE_COMPLETE	Placeholder for extended functionality after the contact use is deleted
CUSTOMER_IMP_CONTACT_USE_LOAD	Extended mapping from the contact use model to the corresponding structure
CUSTOMER_IMP_CONTACT_USE_MODEL_TYPE	Determines the appropriate model based on the contact use structure
CUSTOMER_IMP_CONTACT_USE_SEARCH	
CUSTOMER_IMP_CONTACT_USE_SEARCH_COMPLETE	
CUSTOMER_IMP_CONTACT_USE_STORE	Extended mapping from the contact use structure to the corresponding model
CUSTOMER_IMP_CONTACT_USE_STORE_COMPLETE	Placeholder for extended functionality after the contact use is stored
CUSTOMER_IMP_CONTACT_USE_STRUCTURE_TYPE	Determines the appropriate structure based on the contact use model

16.2.6 Customer Interaction

The customer interaction object captures information about a specific customer interaction, including the interaction type and subtype, note, attachment URL, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_CUSTOMER_INTERACTION_DELETE_COMPLETE	Placeholder for extended functionality after customer interaction is deleted
CUSTOMER_IMP_CUSTOMER_INTERACTION_LOAD	Extended mapping from the customer interaction model to the corresponding structure
CUSTOMER_IMP_CUSTOMER_INTERACTION_MODEL_TYPE	Determines the appropriate model based on the customer interaction structure
CUSTOMER_IMP_CUSTOMER_INTERACTION_SEARCH	Returns a DataObjectList of customer interactions matching the search criteria
CUSTOMER_IMP_CUSTOMER_INTERACTION_SEARCH_COMPLETE	
CUSTOMER_IMP_CUSTOMER_INTERACTION_STORE	Extended mapping from the customer interaction structure to the corresponding model
CUSTOMER_IMP_CUSTOMER_INTERACTION_STORE_COMPLETE	Placeholder for extended functionality after the customer interaction is stored
CUSTOMER_IMP_CUSTOMER_INTERACTION_STRUCTURE_TYPE	Determines the appropriate structure based

Event Name	Description
	on the customer interaction model

16.2.7 External Identifier

The external identifier object captures the information about the object's alias in another system. This object is not extendible.

16.2.8 Identification

The identification object captures information about a specific identification, including identification type and number, valid from date, valid to date, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_IDENTIFICATION_DELETE_COMPLETE	Placeholder for extended functionality after the identification is deleted.
CUSTOMER_IMP_IDENTIFICATION_LOAD	Extended mapping from the identification model to the corresponding structure.
CUSTOMER_IMP_IDENTIFICATION_MASK	Determines whether masking is required. It returns null if it not required; otherwise, it returns the masked value.
CUSTOMER_IMP_IDENTIFICATION_MODEL_TYPE	Determines the appropriate model based on the identification structure.
CUSTOMER_IMP_IDENTIFICATION_SEARCH	Returns a DataObjectList of identifications matching the search criteria.
CUSTOMER_IMP_IDENTIFICATION_SEARCH_COMPLETE	
CUSTOMER_IMP_IDENTIFICATION_STORE	Extended mapping from the identification structure to the corresponding model.
CUSTOMER_IMP_IDENTIFICATION_STORE_COMPLETE	Placeholder for extended functionality after the identification is stored.
CUSTOMER_IMP_IDENTIFICATION_STRUCTURE_TYPE	Determines the appropriate structure based on the identification model.

16.2.9 Party

The party object captures information about a specific party, including entity type, language, date of birth, date of death, gender, industry, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	**Description**
CUSTOMER_IMP_PARTY_DELETE_COMPLETE	Placeholder for extended functionality after the party is deleted
CUSTOMER_IMP_PARTY_LOAD	Extended mapping from the party model to the corresponding structure
CUSTOMER_IMP_PARTY_MODEL_TYPE	Determines the appropriate model based on the party structure
CUSTOMER_IMP_PARTY_SEARCH	Returns a DataObjectList of parties matching the search criteria
CUSTOMER_IMP_PARTY_SEARCH_COMPLETE	
CUSTOMER_IMP_PARTY_STORE	Extended mapping from the party structure to the corresponding model
CUSTOMER_IMP_PARTY_STORE_COMPLETE	Placeholder for extended functionality after the party is stored
CUSTOMER_IMP_PARTY_STRUCTURE_TYPE	Determines the appropriate structure based on the party model

16.2.10 Party Name

The party name object captures information about a specific party name, including type, family generation, first name, form of address, formatted, last name, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
CUSTOMER_IMP_PARTY_NAME_DELETE_COMPLETE	Placeholder for extended functionality after the party name is deleted
CUSTOMER_IMP_PARTY_NAME_LOAD	Extended mapping from the party name model to the corresponding structure
CUSTOMER_IMP_PARTY_NAME_MODEL_TYPE	Determines the appropriate model based on the party name structure
CUSTOMER_IMP_PARTY_NAME_STORE	Extended mapping from the party name structure to the corresponding model
CUSTOMER_IMP_PARTY_NAME_STORE_COMPLETE	Placeholder for extended functionality after the party name is stored
CUSTOMER_IMP_PARTY_NAME_STRUCTURE_TYPE	Determines the appropriate structure based on the party name model

16.3 API Event Handlers

16.3.1 CUSTOMER_CUSTOMER_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a customer and hierarchically creates all its associated entities (contact mediums, contacts, accounts, party) by integrating its corresponding API event handlers: • CUSTOMER_PARTY_CREATE • CUSTOMER_PARTY_GET • CUSTOMER_CONTACT_MEDIUM_CREATE • CUSTOMER_CONTACT_CREATE • CUSTOMER_EXTERNAL_IDENTIFIER_CREATE • CUSTOMER_ACCOUNT_CREATE When creating a customer in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CUSTOMER_MODEL_TYPE • CUSTOMER_IMP_CUSTOMER_STORE • CUSTOMER_IMP_CUSTOMER_STORE_COMPLETE • CUSTOMER_IMP_CUSTOMER_STRUCTURE_TYPE • CUSTOMER_IMP_CUSTOMER_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.2 CUSTOMER_CUSTOMER_GET

Event ID	Functional Description
CW	The implementation of this event retrieves customer details and hierarchically retrieves all its associated entities (contact mediums, contacts, accounts, party) by integrating its corresponding API event handlers: • CUSTOMER_PARTY_GET • CUSTOMER_CONTACT_MEDIUM_GET • CUSTOMER_CONTACT_GET • CUSTOMER_EXTERNAL_IDENTIFIER_GET • CUSTOMER_ACCOUNT_GET When retrieving the detail of the customer entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CUSTOMER_MODEL_TYPE • CUSTOMER_IMP_CUSTOMER_STRUCTURE_TYPE • CUSTOMER_IMP_CUSTOMER_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.3 CUSTOMER_CUSTOMER_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of a customer. When removing a customer in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CUSTOMER_MODEL_TYPE • CUSTOMER_IMP_CUSTOMER_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.4 CUSTOMER_CUSTOMER_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for customers satisfying the search criterion. When searching for a customer in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CUSTOMER_MODERN_SEARCH • CUSTOMER_IMP_CUSTOMER_SEARCH • CUSTOMER_IMP_CUSTOMER_SEARCH_COMPLETE • CUSTOMER_IMP_CUSTOMER_STRUCTURE_TYPE • CUSTOMER_IMP_CUSTOMER_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.5 CUSTOMER_CUSTOMER_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing customer and its party by integrating its corresponding API event handler: • CUSTOMER_PARTY_UPDATE When updating a customer in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CUSTOMER_MODEL_TYPE • CUSTOMER_IMP_CUSTOMER_STORE • CUSTOMER_IMP_CUSTOMER_STORE_COMPLETE • CUSTOMER_IMP_CUSTOMER_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.6 CUSTOMER_CONTACT_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a contact and hierarchically creates all its associated entities (contact mediums, contact use, party) by integrating its corresponding API event handlers: • CUSTOMER_PARTY_CREATE • CUSTOMER_PARTY_GET • CUSTOMER_CONTACT_MEDIUM_CREATE • CUSTOMER_CONTACT_USE_CREATE When creating a contact in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MODEL_TYPE • CUSTOMER_IMP_CONTACT_STORE • CUSTOMER_IMP_CONTACT_STORE_COMPLETE • CUSTOMER_IMP_CONTACT_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.7 CUSTOMER_CONTACT_GET

Event ID	Functional Description
CW	The implementation of this event retrieves details of a contact and hierarchically retrieves all its associated entities (contact mediums, contact use, party) by integrating its corresponding API event handlers: • CUSTOMER_PARTY_GET • CUSTOMER_CONTACT_MEDIUM_GET • CUSTOMER_CONTACT_USE_GET When retrieving contact entity details, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MODEL_TYPE • CUSTOMER_IMP_CONTACT_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.8 CUSTOMER_CONTACT_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of a contact. When removing a contact in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MODEL_TYPE • CUSTOMER_IMP_CONTACT_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.9 CUSTOMER_CONTACT_QUERY

Event ID	Functional Description
CW	The implementation of this event search for contacts satisfying the search criterion. When searching for a contact in the application, the following implementation events are integrated to support extendibility:

	• CUSTOMER_IMP_CONTACT_SEARCH • CUSTOMER_IMP_CONTACT_SEARCH_COMPLETE • CUSTOMER_IMP_CONTACT_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.10 CUSTOMER_CONTACT_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing contact and its party by integrating its corresponding API event handler: • CUSTOMER_PARTY_UPDATE • CUSTOMER_PARTY_CREATE When updating a contact in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MODEL_TYPE • CUSTOMER_IMP_CONTACT_STORE • CUSTOMER_IMP_CONTACT_STORE_COMPLETE • CUSTOMER_IMP_CONTACT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.11 CUSTOMER_CONTACT_MEDIUM_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a contact medium. When creating contact medium in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MEDIUM_MODEL_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_STORE • CUSTOMER_IMP_CONTACT_MEDIUM_STORE_COMPLETE • CUSTOMER_IMP_CONTACT_MEDIUM_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.12 CUSTOMER_CONTACT_MEDIUM_GET

Event ID	Functional Description
CW	The implementation of this event retrieves details of a contact medium. When retrieving details of the contact medium entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MEDIUM_MODEL_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.13 CUSTOMER_CONTACT_MEDIUM_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of a contact medium. When removing a contact medium in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MEDIUM_MODEL_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.14 CUSTOMER_CONTACT_MEDIUM_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for contact mediums satisfying the search criterion. When searching for a contact medium in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MEDIUM_SEARCH • CUSTOMER_IMP_CONTACT_MEDIUM_SEARCH_COMPLETE • CUSTOMER_IMP_CONTACT_MEDIUM_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.15 CUSTOMER_CONTACT_MEDIUM_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing contact medium. When updating a contact medium in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_MEDIUM_MODEL_TYPE • CUSTOMER_IMP_CONTACT_MEDIUM_STORE • CUSTOMER_IMP_CONTACT_MEDIUM_STORE_COMPLETE • CUSTOMER_IMP_CONTACT_MEDIUM_LOAD This implementation may return the following error codes:

Event ID	Functional Description
	• COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.16 CUSTOMER_CONTACT_USE_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a contact use. When creating contact use in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_USE_MODEL_TYPE • CUSTOMER_IMP_CONTACT_USE_STORE • CUSTOMER_IMP_CONTACT_USE_STORE_COMPLETE • CUSTOMER_IMP_CONTACT_USE_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_USE_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.17 CUSTOMER_CONTACT_USE_GET

Event ID	Functional Description
CW	The implementation of this event retrieves details of a contact use. When retrieving details of the contact use entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_USE_MODEL_TYPE • CUSTOMER_IMP_CONTACT_USE_STRUCTURE_TYPE • CUSTOMER_IMP_CONTACT_USE_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.18 CUSTOMER_CONTACT_USE_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of a contact use. When removing a contact use in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_USE_MODEL_TYPE • CUSTOMER_IMP_CONTACT_USE_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.19 CUSTOMER_CONTACT_USE_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing contact use. When updating a contact use in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_CONTACT_USE_MODEL_TYPE • CUSTOMER_IMP_CONTACT_USE_STORE • CUSTOMER_IMP_CONTACT_USE_STORE_COMPLETE • CUSTOMER_IMP_CONTACT_USE_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.20 CUSTOMER_INTERACTION_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a customer interaction. When creating a customer interaction in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_INTERACTION_MODEL_TYPE • CUSTOMER_IMP_INTERACTION_STORE • CUSTOMER_IMP_INTERACTION_STORE_COMPLETE • CUSTOMER_IMP_INTERACTION_STRUCTURE_TYPE • CUSTOMER_IMP_INTERACTION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.21 CUSTOMER_INTERACTION_GET

Event ID	Functional Description
CW	The implementation of this event retrieves details of a customer interaction. When retrieving details of the customer interaction entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_INTERACTION_MODEL_TYPE • CUSTOMER_IMP_INTERACTION_STRUCTURE_TYPE • CUSTOMER_IMP_INTERACTION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.22 CUSTOMER_INTERACTION_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of a customer interaction. When removing a customer interaction in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_INTERACTION_MODEL_TYPE • CUSTOMER_IMP_INTERACTION_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.23 CUSTOMER_INTERACTION_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for customer interactions satisfying the search criterion. When searching for a customer interaction in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_INTERACTION_SEARCH • CUSTOMER_IMP_INTERACTION_SEARCH_COMPLETE • CUSTOMER_IMP_INTERACTION_STRUCTURE_TYPE • CUSTOMER_IMP_INTERACTION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.24 CUSTOMER_INTERACTION_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing customer interaction. When updating a customer interaction in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_INTERACTION_MODEL_TYPE • CUSTOMER_IMP_INTERACTION_STORE • CUSTOMER_IMP_INTERACTION_STORE_COMPLETE • CUSTOMER_IMP_INTERACTION_LOAD

Event ID	Functional Description
	This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.25 CUSTOMER_ACCOUNT_CREATE

Event ID	Functional Description
CW	The implementation of this event creates an account and hierarchically creates all its associated entities (contact use, external identifiers, sub accounts) by integrating its corresponding API event handlers: • CUSTOMER_CONTACT_USE_CREATE • CUSTOMER_EXTERNAL_IDENTIFIER_CREATE When creating account in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_ACCOUNT_MODEL_TYPE • CUSTOMER_IMP_ACCOUNT_STORE • CUSTOMER_IMP_ACCOUNT_STORE_COMPLETE • CUSTOMER_IMP_ACCOUNT_STRUCTURE_TYPE • CUSTOMER_IMP_ACCOUNT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.26 CUSTOMER_ACCOUNT_GET

Event ID	Functional Description
CW	The implementation of this event retrieves details of an account and hierarchically retrieves all its associated entities (contact use, external identifiers, sub accounts) by integrating its corresponding API event handlers: • CUSTOMER_CONTACT_USE _GET • CUSTOMER_EXTERNAL_IDENTIFIER _GET When retrieving details of the account entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_ACCOUNT_MODEL_TYPE • CUSTOMER_IMP_ACCOUNT_STRUCTURE_TYPE • CUSTOMER_IMP_ACCOUNT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.27 CUSTOMER_ACCOUNT_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of an account. When removing an account in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_ACCOUNT_MODEL_TYPE • CUSTOMER_IMP_ACCOUNT_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.28 CUSTOMER_ACCOUNT_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for accounts satisfying the search criterion. When searching for an account in the application, the following implementation Events are integrated to support extendibility: • CUSTOMER_IMP_ACCOUNT_SEARCH • CUSTOMER_IMP_ACCOUNT_SEARCH_COMPLETE • CUSTOMER_IMP_ACCOUNT_STRUCTURE_TYPE • CUSTOMER_IMP_ACCOUNT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.29 CUSTOMER_ACCOUNT_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing account. When updating an account in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_ACCOUNT_MODEL_TYPE • CUSTOMER_IMP_ACCOUNT_STORE • CUSTOMER_IMP_ACCOUNT_STORE_COMPLETE • CUSTOMER_IMP_ACCOUNT_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.30 CUSTOMER_IDENTIFICATION_CREATE

Event ID	Functional Description
CW	The implementation of this event creates customer identification. When creating customer identification in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_IDENTIFICATION_MODEL_TYPE • CUSTOMER_IMP_IDENTIFICATION_MASK • CUSTOMER_IMP_IDENTIFICATION_STORE • CUSTOMER_IMP_IDENTIFICATION_STORE_COMPLETE • CUSTOMER_IMP_IDENTIFICATION_STRUCTURE_TYPE • CUSTOMER_IMP_IDENTIFICATION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.31 CUSTOMER_IDENTIFICATION_GET

Event ID	Functional Description
CW	The implementation of this event retrieves customer identification details. When retrieving details of the customer identification entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_IDENTIFICATION_MODEL_TYPE • CUSTOMER_IMP_IDENTIFICATION_STRUCTURE_TYPE • CUSTOMER_IMP_IDENTIFICATION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.32 CUSTOMER_IDENTIFICATION_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of customer identification. When removing customer identification in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_IDENTIFICATION_MODEL_TYPE • CUSTOMER_IMP_IDENTIFICATION_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.33 CUSTOMER_IDENTIFICATION_QUERY

Event ID	Functional Description
CW	The implementation of this event searches for customer identification satisfying the search criterion. When searching for customer identification in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_IDENTIFICATION_SEARCH • CUSTOMER_IMP_IDENTIFICATION_SEARCH_COMPLETE • CUSTOMER_IMP_IDENTIFICATION_STRUCTURE_TYPE • CUSTOMER_IMP_INTERACTION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.34 CUSTOMER_IDENTIFICATION_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates existing customer identification. When updating customer identification in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_IDENTIFICATION_MODEL_TYPE • CUSTOMER_IMP_IDENTIFICATION_MASK • CUSTOMER_IMP_IDENTIFICATION_STORE • CUSTOMER_IMP_IDENTIFICATION_STORE_COMPLETE

Event ID	Functional Description
	• CUSTOMER_IMP_IDENTIFICATION_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.35 CUSTOMER_PARTY_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a party and hierarchically creates all its associated entities (party name, identification) by integrating to its corresponding API event handlers: • CUSTOMER_PARTY_NAME_CREATE • CUSTOMER_ IDENTIFICATION _CREATE When creating a party in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_MODEL_TYPE • CUSTOMER_IMP_PARTY_STORE • CUSTOMER_IMP_PARTY_STORE_COMPLETE • CUSTOMER_IMP_PARTY_STRUCTURE_TYPE • CUSTOMER_IMP_PARTY_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.36 CUSTOMER_PARTY_GET

Event ID	Functional Description
CW	The implementation of this event retrieves party details and hierarchically retrieves all its associated entities (party name, identification) by integrating its corresponding API event handlers: • CUSTOMER_PARTY_NAME_GET • CUSTOMER_ IDENTIFICATION _GET When retrieving details of the party entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_MODEL_TYPE • CUSTOMER_IMP_PARTY_STRUCTURE_TYPE • CUSTOMER_IMP_PARTY_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.37 CUSTOMER_PARTY_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes party details. When removing a party in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_MODEL_TYPE • CUSTOMER_IMP_PARTY_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.38 CUSTOMER_PARTY_QUERY

Event ID	Functional Description
CW	The implementation of this event search for parties satisfying the search criterion. When searching for a party in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_SEARCH • CUSTOMER_IMP_PARTY_SEARCH_COMPLETE • CUSTOMER_IMP_PARTY_STRUCTURE_TYPE • CUSTOMER_IMP_PARTY_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.39 CUSTOMER_PARTY_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing party and its party name by integrating to its corresponding API event handler: • CUSTOMER_PARTY_NAME_UPDATE When updating a party in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_MODEL_TYPE • CUSTOMER_IMP_PARTY_STORE • CUSTOMER_IMP_PARTY_STORE_COMPLETE • CUSTOMER_IMP_PARTY_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.3.40 CUSTOMER_PARTY_NAME_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a party name. When creating a party name in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_NAME_MODEL_TYPE • CUSTOMER_IMP_PARTY_NAME_STORE • CUSTOMER_IMP_PARTY_NAME_STORE_COMPLETE • CUSTOMER_IMP_PARTY_NAME_STRUCTURE_TYPE • CUSTOMER_IMP_PARTY_NAME_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0003 • COM_ERR_0004

16.3.41 CUSTOMER_PARTY_NAME_GET

Event ID	Functional Description
CW	The implementation of this event retrieves details of a party name. When retrieving details of the party name entity, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_NAME_MODEL_TYPE • CUSTOMER_IMP_PARTY_NAME_STRUCTURE_TYPE • CUSTOMER_IMP_PARTY_NAME_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003

16.3.42 CUSTOMER_PARTY_NAME_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes details of a party name. When removing a party name in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_NAME_MODEL_TYPE • CUSTOMER_IMP_PARTY_NAME_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0005

16.3.43 CUSTOMER_PARTY_NAME_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing party name. When updating a party name in the application, the following implementation events are integrated to support extendibility: • CUSTOMER_IMP_PARTY_NAME_MODEL_TYPE • CUSTOMER_IMP_PARTY_NAME_STORE • CUSTOMER_IMP_PARTY_NAME_STORE_COMPLETE • CUSTOMER_IMP_PARTY_NAME_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002 • COM_ERR_0003 • COM_ERR_0004

16.4 Implementation Event Handlers

The Customer module implementation provides handling to some implementation-defined events.

16.4.1 CUSTOMER_IMP_ACCOUNT_SEARCH

See *Implementation Events* for a detailed description.

Event ID	Functional Description
CW	Look up accounts matching all the following search criteria when provided: • Customer ID • Account Name • Active • Account Type • Account Status • Archive • Primary • Start Activity Date • End Activity Date • Parent Account ID

16.4.2 CUSTOMER_IMP_CONTACT_MEDIUM_SEARCH

See *Implementation Events* for a detailed description.

Event ID	Functional Description
CW	Look up contact mediums matching all the following search criteria when provided: • Type • Value • Valid From Date • Valid To Date • Contact Name • Party Role ID • Primary

16.4.3 CUSTOMER_IMP_CONTACT_SEARCH

See *Implementation Events* for a detailed description.

Event ID	Functional Description
CW	Look up contacts matching all the following search criteria when provided: • Customer ID • Account ID • Site ID • Primary

16.4.4 CUSTOMER_IMP_CUSTOMER_MODERN_SEARCH

Event ID	Functional Description
CW	Look up records for any entities matching any part of the search key. Entities that are considered include: • ACCOUNT • PARTY • CUSTOMER

16.4.5 CUSTOMER_IMP_CUSTOMER_SEARCH

Event ID	Functional Description
CW	Look up customers matching all the following search criteria when provided: • Entity Type • First Name • Last Name • Customer Status • Phone Contact Medium • Email Contact Medium • Birthdate • Mother’s Maiden Name • Customer External Identifier • Account External Identifier • Company Name • Company Title • Company Industry • Company Registration Date

16.4.6 CUSTOMER_IMP_IDENTIFICATION_SEARCH

See *Implementation Events* for a detailed description.

Event ID	Functional Description
CW	Look up identifications matching all the following search criteria when provided: • Party ID • Identification Type • Valid From • Identification Number • Credit Card type • Credit Card Expiry

16.4.7 CUSTOMER_IMP_INTERACTION_SEARCH

See *Implementation Events* for a detailed description.

Event ID	Functional Description
CW	Look up customer interactions matching all the following search criteria when provided: • Customer Interaction ID • Customer Interaction Date From • Customer Interaction Date To • Customer Interaction Type • Customer Interaction Sub Type • Customer ID • Account ID • Site ID • Contact ID • Order ID • Ticket ID

16.4.8 CUSTOMER_IMP_PARTY_SEARCH

See *Implementation Events* for a detailed description.

Event ID	Functional Description
CW	Look up parties matching all the following search criteria when provided: • First Name • Last Name • Entity Type • Birth Date • Mother’s Maiden Name • Trading Name • Company Registration Date • Company Industry • Company Organization Type • Company Revenue Range

16.5 Error Codes

16.5.1 CUST_ERR_0001

IdentificationNumber and verificationNumber do not match.

17 Site Model

17.1 About This Section

This section will provide application developers with an understanding of extending the Site model.

17.2 Site Model

The site model is designed to capture site information used and maintained by application.

17.2.1 Site

Site object captures information about a specific site, including the site name, start of activity date, end of activity date, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
SITE_IMP_SITE_SEARCH	Returns a DataObjectList of sites matching search criteria
SITE_IMP_SITE_SEARCH_COMPLETE	
SITE_IMP_SITE_LOAD	Extended mapping from site model to corresponding structure
SITE_IMP_SITE_STORE	Extended mapping from site structure to corresponding model
SITE_IMP_SITE_STORE_COMPLETE	Placeholder for extended functionality after site is stored
SITE_IMP_SITE_DELETE_COMPLETE	Placeholder for extended functionality after site is deleted
SITE_IMP_SITE_MODEL_TYPE	Determines the appropriate model based on the site structure
SITE_IMP_SITE_STRUCTURE_TYPE	Determines the appropriate structure based on the site model

17.2.2 Serviceability

Serviceability object captures information about a specific serviceability, including technology type, inside wiring, CLLI (Common Language Location Identifier), horizontal/vertical coordinates, and so on. For extendibility, the following events are available for applications to extend the functionalities provided by standard implementation.

Event Name	Description
SITE_IMP_SERVICEABILITY_LOAD	Extended mapping from serviceability model to corresponding structure
SITE_IMP_SERVICEABILITY_STORE	Extended mapping from serviceability structure to corresponding model
SITE_IMP_SERVICEABILITY_STORE_COMPLETE	Placeholder for extended functionality after serviceability is stored
SITE_IMP_SERVICEABILITY_DELETE_COMPLETE	Placeholder for extended functionality after serviceability is deleted
SITE_IMP_SERVICEABILITY_MODEL_TYPE	Determines the appropriate model based on the serviceability structure
SITE_IMP_SERVICEABILITY_STRUCTURE_TYPE	Determines the appropriate structure based on the serviceability model

17.2.3 External Identifier

External Identifier is an object used to capture information about the identification of a site by an external system. Extendibility for this object is not supported.

17.3 Site API Event Handlers

The following are event handlers for the Site API.

17.3.1 SITE_SITE_CREATE

Event ID	Functional Description
CW	The implementation of this event creates a site and its external identifiers. When creating site in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SITE_MODEL_TYPE • SITE_IMP_SITE_STORE • SITE_IMP_SITE_STORE_COMPLETE • SITE_IMP_SITE_LOAD This implementation may return the following error codes: • COM_ERR_0001

17.3.2 SITE_SITE_GET

Event ID	Functional Description
CW	The implementation of this event retrieves the detail of a site and its external identifiers. When retrieving site in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SITE_MODEL_TYPE • SITE_IMP_SITE_STRUCTURE_TYPE • SITE_IMP_SITE_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002

17.3.3 SITE_SITE_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes the detail of a site. When removing a site in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SITE_MODEL_TYPE • SITE_IMP_SITE_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001

17.3.4 SITE_SITE_QUERY

Event ID	Functional Description
CW	The implementation of this event search for sites satisfying the search criterion. When searching for a site in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SITE_MODERN_SEARCH • SITE_IMP_SITE_SEARCH • SITE_IMP_SITE_SEARCH_COMPLETE • SITE_IMP_SITE_STRUCTURE_TYPE • SITE_IMP_SITE_LOAD This implementation may return the following error codes: • COM_ERR_0001

17.3.5 SITE_SITE_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing site and its external identifiers. When creating site in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SITE_MODEL_TYPE • SITE_IMP_SITE_STORE • SITE_IMP_SITE_STORE_COMPLETE • SITE_IMP_SITE_STRUCTURE_TYPE • SITE_IMP_SITE_LOAD This implementation may return the following error codes: • COM_ERR_0001

	• COM_ERR_0002

17.3.6 SITE_SERVICEABILITY_CREATE

Event ID	Functional Description
CW	The implementation of this event creates serviceability. When creating serviceability in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SERVICEABILITY_MODEL_TYPE • SITE_IMP_SERVICEABILITY_STORE • SITE_IMP_SERVICEABILITY_STORE_COMPLETE • SITE_IMP_SERVICEABILITY_LOAD This implementation may return the following error codes: • COM_ERR_0001

17.3.7 SITE_SERVICEABILITY_GET

Event ID	Functional Description
CW	The implementation of this event retrieves the detail of serviceability and its external identifiers. When retrieving serviceability in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SERVICEABILITY_MODEL_TYPE • SITE_IMP_SERVICEABILITY_STRUCTURE_TYPE • SITE_IMP_SERVICEABILITY_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002

17.3.8 SITE_SERVICEABILITY_REMOVE

Event ID	Functional Description
CW	The implementation of this event removes the detail of serviceability. When removing a serviceability in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SERVICEABILITY_MODEL_TYPE • SITE_IMP_SERVICEABILITY_DELETE_COMPLETE This implementation may return the following error codes: • COM_ERR_0001

17.3.9 SITE_SERVICEABILITY_UPDATE

Event ID	Functional Description
CW	The implementation of this event updates an existing serviceability and its external identifiers. When creating serviceability in the application, the following Implementation Events are integrated to support extendibility: • SITE_IMP_SERVICEABILITY_MODEL_TYPE • SITE_IMP_SERVICEABILITY_STORE • SITE_IMP_SERVICEABILITY_STORE_COMPLETE • SITE_IMP_SERVICEABILITY_STRUCTURE_TYPE • SITE_IMP_SERVICEABILITY_LOAD This implementation may return the following error codes: • COM_ERR_0001 • COM_ERR_0002

17.4 Implementation Event Handlers

17.4.1 SITE_IMP_SITE_SEARCH

Event ID	Functional Description
CW	Look up addresses matching all of the following search criterions when provided: • Site name (like) • Active • Start of Activity Date • End of Activity Date

17.4.2 SITE_IMP_SITE_MODERN_SEARCH

Event ID	Functional Description
CW	Look up records for any entities matching any part of the search key. Entities that are considered include: • SITE

18 Acronyms

CIM – Configuration Information Management

SID - Shared Information/Data

19 Reference List

The following is a list of documentation for reference:

- *CIM User Guide*
- *System Administration User Guide*
- *System Configuration User Guide*
- *User Profile Administration User Guide*
- *Velocity Studio User Guide*

20 Trademarks

21 Disclaimer