Oracle® Retail Merchandising System

Custom Flex Attribute Solution Implementation Guide

Release 14.1.1

E62017-01

May 2015

ORACLE®

Oracle Retail Merchandising System Custom Flex Attribute Solution Implementation Guide, Release 14.1.1

E62017-01

Primary Author: Chaitra Ramaprasad

reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Send Us Your Comments

Oracle Retail Merchandising System Custom Flex Attribute Solution Implementation Guide, Release 14.1.1

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

This *Custom Flex Attribute Solution Implementation Guide* describes how you can plan, create, and maintain additional attributes using CFAS in existing pre-enabled RMS entities.

Audience

This document is intended for application integrators and implementation personnel who are familiar with the Oracle Forms based development. Knowledge of Oracle Retail Merchandising System (RMS) is also required.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents:

- *Oracle Retail Merchandising System Installation Guide*
- *Oracle Retail Merchandising System Reports User Guide*
- *Oracle Retail Merchandising System User Guide and Online Help*
- *Oracle Retail Merchandising System Release Notes*
- *Oracle Retail Merchandising System Operations Guide*
- *Oracle Retail Merchandising System Data Model*
- *Oracle Retail Merchandising System Data Access Schema Data Model*
- *Oracle Retail Merchandising Security Guide*
- *Oracle Retail Merchandising Implementation Guide*

- *Oracle Retail Merchandising Data Conversion Operations Guide*
- *Oracle Retail Merchandising Batch Schedule*
- *Oracle Retail POS Suite/Merchandising Operations Management Implementation Guide*
- Oracle Retail Sales Audit documentation
- Oracle Retail Trade Management documentation
- Oracle Retail Fiscal Management documentation

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 14.1) or a later patch release (for example, 14.1.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction

Custom Flex Attribute Solution (CFAS) for Oracle Retail Merchandising System (RMS) is a metadata driven framework that enables you to set up additional attributes on the pre-enabled RMS entities without having to change the existing screen or make any changes in the application code.

This chapter provides an introduction to the Custom Flex Attribute Solution (CFAS) highlighting the associated components and data flow. It also provides information on the organization of content in this document and how you can use it. This chapter includes the following topics:

- Road Map for Implementing CFAS
- Getting Started with CFAS

> **Note:** The CFAS framework does not publish new attributes to the follow up systems, such as Oracle Retail Integration Bus, Oracle Retail Stores Inventory Management, or Oracle Retail Warehouse Management System.

Road Map for Implementing CFAS

In order to implement CFAS for production, it is recommended that you review the instructions in this guide in the following order.

Table 1–1 Road Map for Implementing CFAS

Task	Description
Pre-Implementation Tasks	
1.	Familiarize with CFAS concepts described in the current chapter. See the chapter Introduction.
2.	Review the business process flows and plan the attributes carefully based on your requirements. Consider attribute grouping, performance, detail level, and internationalization. For more information, see Planning Custom Flex Attributes and Planning Your Implementation.
Implementation Task	
3.	If you want to start using the pre-enabled entities, proceed to the part CFAS Administration in this book. See the chapter CFAS Maintenance Screens.
Maintenance Tasks	
4.	For more information on the considerations for maintaining and upgrading the flex attributes that are in use, see the chapter Ongoing Maintenance.

Table 1–1 (Cont.) Road Map for Implementing CFAS

Task	Description
Additional References	
	For examples of common RMS customization and how you can leverage the CFAS framework, see the appendix CFAS Use Cases.
	For more information on the CFAS-specific database scripts, see the appendix CFAS Database Scripts.
	For more information on the validation routines in the CFAS user interface, see the appendix CFAS User Interface Validation Routines.
	For more information on debugging CFAS, see the appendix Debugging CFAS.
	For more information on the CFAS table definitions, see the appendix CFAS Table Definitions.

Getting Started with CFAS

Custom Flex Attribute Solution (CFAS) for Oracle Retail Merchandising System (RMS) is a metadata driven framework that can be used to support client-specific customizations.

This section includes the following topics that provides more information on the CFAS framework:

- About CFAS
- CFAS Components
- CFAS Data Flow

About CFAS

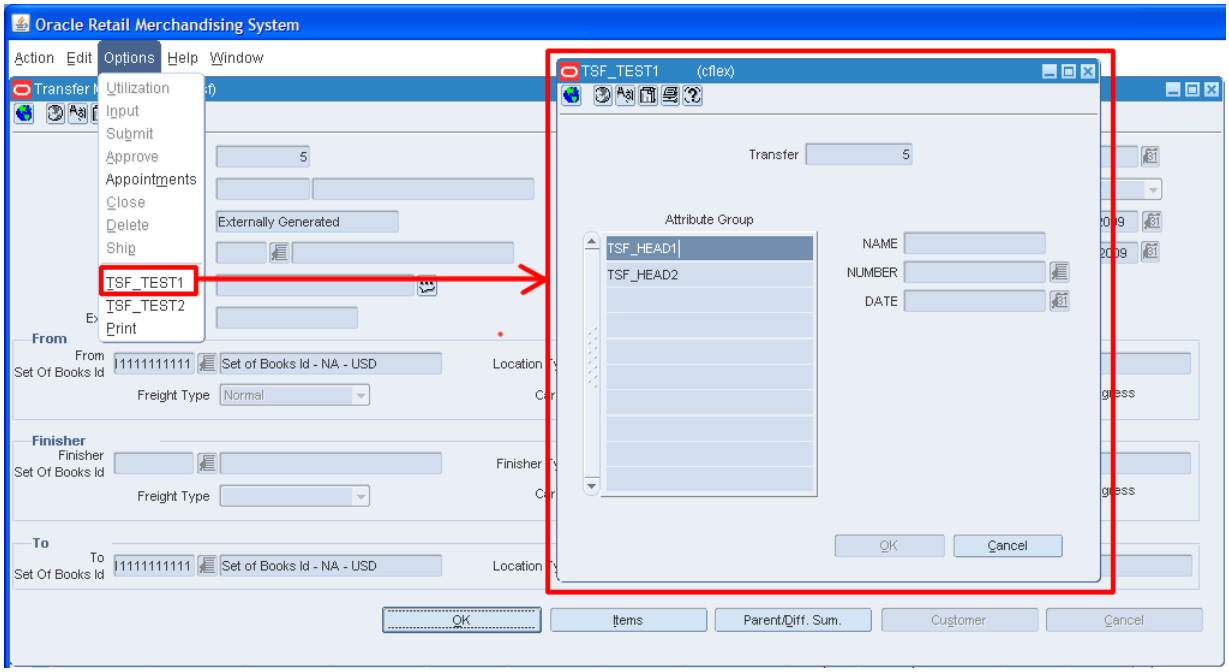
CFAS is designed in such a manner that certain pre-existing form windows can be easily customized to include additional fields or attributes. The CFAS framework enables you to set up additional attributes on existing RMS entities without having to change the existing screen or make any changes in the application code. The additional attributes can be new attributes that are needed to support expanded client functionality or to capture additional information from legacy systems. The CFAS framework also allows the storage and validation of these attributes.

The additional attributes can be accessed using the Options menu in the relevant form windows where they have been enabled. This ensures that the additional attributes do not clutter the existing screen when they have not been implemented or used.

Example: Additional CFAS Attributes in the Transfer Maintenance Form

The following figure illustrates a CFAS user interface that can be accessed by clicking the TSF_TEST1 option from the **Options** menu of the Transfer Maintenance form:

Figure 1–1 Illustration of a CFAS User Interface on Transfer Maintenance Form



In the illustration above,

- The Transfer Maintenance form is an entity that has been extended to use the CFAS framework.
- Additional attributes for the Transfer Maintenance form can be accessed using the TSF_TEST1 menu option from the Options menu.
- In the TSF_TEST1 screen, the NAME, NUMBER, and DATE fields correspond to individual CFAS attributes.
- These attributes belong to the TSF_HEAD1 attribute group. Each attribute group can include up to 22 such attributes including the header fields. For example, if the attribute group has 4 header fields, only 18 attributes can be under each attribute group.
- TSF_HEAD1 and TSF_HEAD2 attribute groups belong to the TSF_TEST1 attribute group set. Attribute group sets appear as menu options in the Options menu and can include many attribute groups.

For more information on these attributes, see Planning Custom Flex Attributes.

CFAS Components

The CFAS framework includes the following components:

- CFAS Maintenance Screens – Available only to the users with the CFAS Administrator role, these group of screens enable you to set up the attributes for a relevant functional area in RMS. For more information, see CFAS Maintenance Screens.

- CFAS Extension User Interface and User Interface (UI) Libraries – The CFAS framework provides a highly configurable metadata-driven user interface built using Oracle Forms. It also includes CFAS UI libraries built specifically to handle metadata. When it is set up, the CFAS UI will be accessible from the Options menu of the relevant RMS form. Users will be able to view extension attributes, capture, or store the relevant information in the CFAS user interface. For more information, see CFAS User Interface Validation Routines.
- CFAS Database Objects – The CFAS framework is driven by information stored in the following CFAS-specific database objects:
  - Extension Installation Tables – These database tables contain information on the extended entities in RMS.
  - Metadata Tables – These database tables contain all the information required to display and capture actual data on the extended entities.
  - CFAS Extension Storage Tables (Entity-specific) – By default, RMS data model will include such entity specific CFAS storage database tables for the pre-enabled entities. Each time you extend a business entity (other than the pre-enabled ones) for customization, create a relevant entity specific CFAS storage table by running the cfagen.ksh batch script.
  - Supporting Custom Objects – The CFAS framework uses the following supporting custom objects that are generated based on the definitions in the Extension Installation and CFAS Metadata tables. These objects are business representation of the data that is to be stored in the generic CFAS extension tables:
    - CFAS Access Views
    - CFAS Staging Tables

    Note: Staging tables are optional.

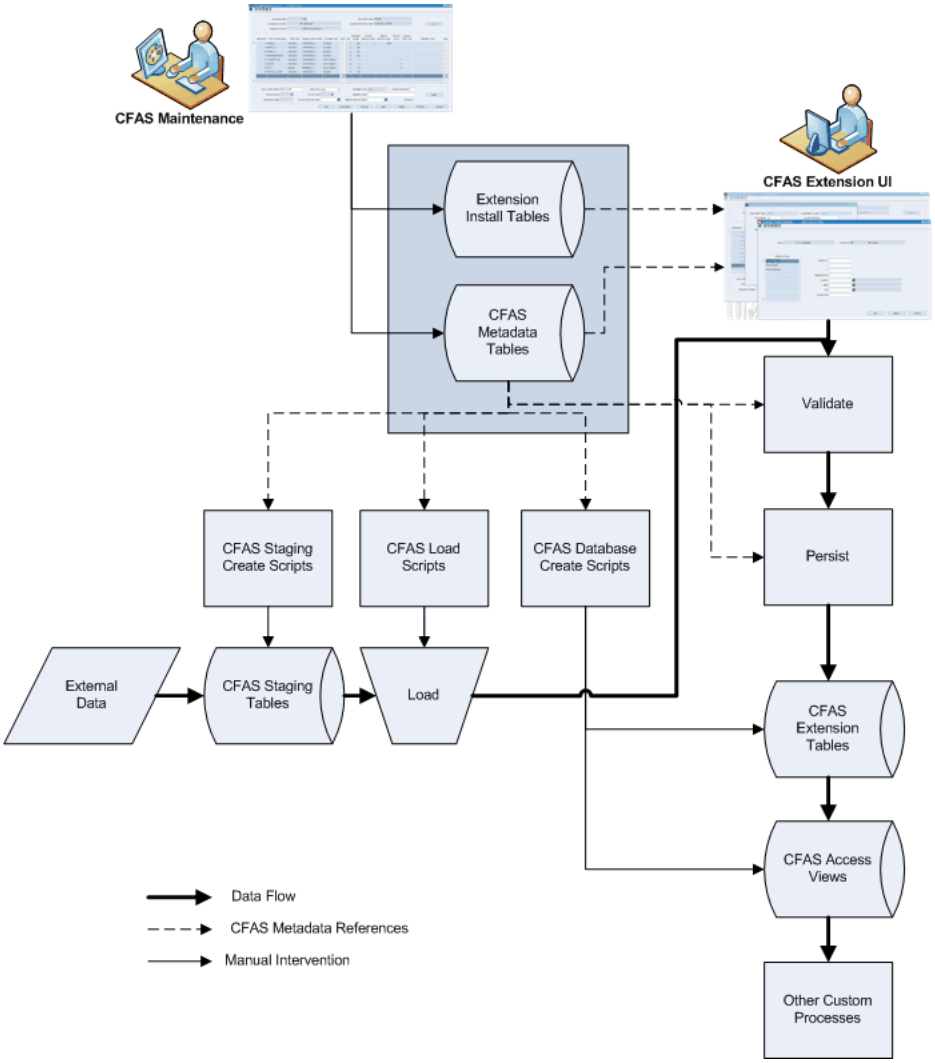
    For more information, see CFAS Table Definitions.
- CFAS Scripts – Along with the user interface and database objects, the CFAS framework also includes the following scripts:
  - CFAS Database Create Script (cfagen.ksh) – Once the attributes are set up in the CFAS Maintenance screens, you must run this script to make or activate the CFAS Extension User Interface for the relevant RMS functional area.
  - CFAS Load Script (cfastgload.ksh) – This script enables you to bulk load the attributes from the CFAS Staging table to the CFAS Extension Storage tables.

    For more information, see CFAS Database Scripts.

CFAS Data Flow

The following figure displays the data flow in a CFAS framework:

Figure 1–2 CFAS Data Flow



The ***Extension Installation Tables*** define the entities and tables that can be customized using the CFAS framework. By default, only certain entities are enabled for customization with initial entries already defined out of the box. Data can be added to these tables using a simple seed data script. In case you want to extend entities other than the ones already defined, you must use the ***CFAS Maintenance*** screens.

The ***CFAS Metadata Tables*** define the business extensions for your implementation. These tables are used to drive the ***CFAS Extension UI*** that will be available for end users and store the data entered. These tables are part of the base RMS data model and include information defined based on your implementation.

Note: The ***Extension Installation Objects*** and ***CFAS Metadata Tables*** are maintained using the ***CFAS Maintenance*** screens.

The ***CFAS Extension UI*** is installed as an additional window on the relevant enabled form. This user interface is a reusable component that reads from the ***CFAS Metadata Tables*** to display the captured attribute data for the entity in an user-friendly way.

To ensure that the data in the user interface are accurate, the user interface automatically includes validation rules based on the type of attribute set up. As an end user enters relevant values in the fields on the ***CFAS Extension UI***, the relevant validation rules defined in the ***CFAS Metadata Tables*** are called out. Validated data is then written to the ***CFAS Extension Storage Tables***. You can choose to define a custom validation function and reference it when you set up an entity, attribute group set, attribute group, or attribute.

The ***CFAS Extension Storage Tables*** are used to store the data entered in the ***CFAS Extension UI*** or loaded from the metadata generated staging tables. These tables are created as part of the process of implementing the CFAS framework for an entity.

The ***CFAS Staging Tables*** enables you to import bulk data from external sources to the ***CFAS Extension Tables***. These tables are created using scripts that function based on the metadata definitions.

The CFAS Load Script (cfastgload.ksh) can then be used to load the data from the ***CFAS Staging Tables*** to the ***CFAS Extension Tables***. This script includes procedures that use the metadata definitions to properly insert data to the ***CFAS Extension Tables***.

CFAS Access Views represent the information stored in the ***CFAS Extension Tables***. These views enable you to easily access data or make the data available to custom processes. The views are generated by scripts. For more information, see CFAS Access Views.

Note: The ***CFAS Access Views*** and ***CFAS Staging Tables*** are automatically created when you run the ***CFAS Database Create Scripts*** (cfagen.ksh) after setting up the ***Extension Installation Objects*** and ***CFAS Metadata Tables***.

2

Planning Custom Flex Attributes

Adding attributes to different entities within RMS is one of the most common customizations. Some attributes are added just for reporting purposes, and some for integrating with other systems, and some for driving processing in RMS. This chapter describes how you can plan and organize the CFAS attributes. It also provides information on the process for creating attributes during an implementation and maintaining attributes when they have been implemented. It includes the following sections:

- Entities
- Attribute Group Sets
- Attribute Groups
- Attributes
- Other Setup Data
- Custom Functions

Before you started adding attributes in the system, you must first set up a plan to structure the attributes and determine how they will be created. When structuring the attributes, you must ensure that they are organized in a manner that makes sense from a business process perspective. For example, combining the attribute groups and attributes together that will be maintained by the same resources in to the same attribute group sets. You must also organize it in such a way that allows for future growth and updates of attributes based on the business changes, especially since there are limits at the group set and attribute level.

Entities

Entities are the functional areas in RMS with which the CFAS is associated. In the current release, the following functionalities (entities) are pre-enabled and support addition of new attributes:

- Supplier
- Store
- Physical Warehouse
- Virtual Warehouse
- Department
- Class
- Subclass

- Item (includes access from main Item dialog as well as Quick Item Add)
- Item/Location
- Item/Supplier
- Item/Supplier/Country
- Item/Supplier/Country/Location
- Address
- ELC Components
- VAT Codes
- Purchase Order Header
- RTV Header
- Deals Header
- Transfer Header

Users can access the CFAS user interface for these pre-enabled entities using the Options menu from the relevant form window. A single menu option is included for each attribute group set associated with the entity. The screens for each of these pre-enabled entities also includes some validation logic to ensure that the required information is entered before opening the attribute screens (for example, for orders, the supplier and dates information must be entered before opening the CFAS user interface). It will also include another validation logic to check the mode (Edit or View) in which the CFAS user interface must appear. This is based on the mode from the relevant form window from where the CFAS user interface is accessed.

Note: Additional customization is required when there is a need to have a particular attribute group accessed in a different manner (for example, if an order is approved, access the group set's attributes in view mode).

When planning the entities, in order to properly validate and display the information in the screens, you must determine the following for each entity:

- **Validation Function** - Validates the information entered before the users save and close the CFAS screen. A validation function is not mandatory and may be needed when further validation is required beyond that set at the attribute level. In such a case, you will need to write a custom package function for the validation function based on your business need.
- **Labels** - Sets the label names for the relevant columns. You must set at least one label for the default language for each label you create. You can choose to create labels for more (alternate) languages based on your business need.
- Description Code - The function that will display the description of the relevant key. For example, SUPPS_SQL.GET_SUPP_NAME will retrieve the supplier name.

Attribute Group Sets

Once you determine the entities that must have the custom flex attributes set up, the next step in the planning process is to determine the attribute group sets required for each entity.

Attribute group sets are at the top level of the flex attribute hierarchy and this is the level that appears in the Options menu for each entity. For each entity, you can define up to 10 attribute group sets.

When planning the attribute group sets, in order to properly validate and display the information in the screens, you must determine the following for each group set:

- **Display Order** - The order in which the attribute group set will appear in the Options menu of the relevant entity.
- **View Name** - The name that will be used for the database view that will be created for this group set. A view is a required information for each group set and is used to facilitate querying attribute information for the group set.
- **Staging Table Name** - The name that will be used for the staging table that will be created for this group set. Although staging table information is optional, it is recommended that you provide staging table name to support data loading from integrated system in the future. In case a staging table is defined, each attribute group set can have one staging table defined and mentioned in the Attribute Group Set Maintenance form. Once activation process is complete, you cannot update the staging table information from the user interface.
- **Qualifier Function** - Validates whether all the required information has been entered on the entity screen from where the CFAS screen is accessed. This occurs before the CFAS screen opens. A qualifier function is not mandatory, but may be required when a special validation is needed beyond ensuring that the primary key is entered. In such a case, you will need to write a custom package function for the qualifier function based on your business need.

  For more information, see Custom Functions.
- **Default Function** - Sets the default values for the attributes in the group set when the CFAS screen opens. A default function is not mandatory and you will need to write a custom package function based on your business need.
- **Validation Function** - Validates the information entered before the users save and close the CFAS screen. A validation function is not mandatory and may be needed when further validation is required beyond that set at the attribute level. In such a case, you will need to write a custom package function for the validation function based on your business need.
- **Labels** - Sets the business name for the group set. The label is the title that will appear in the Option menu for the entity when the attribute group set is accessed. You must set at least one label for the default language for each group set you create. You can choose to create labels for more (alternate) languages based on your business need.

Attribute Groups

Once you determine the attribute group sets needed for each entity, the next step is to determine how the attributes will be organized within these sets. The attributes themselves are organized into groups, which is the middle layer of the CFAS hierarchy. Although you can create as many attribute groups as you want for each group set, you can only have 22 attributes in each attribute group.

> **Note:** The forms user interface, by design, has a limit of 18 lines in a window. Since the entities already include a set number of entity keys, you will only be able to set up less than 18 attributes per attribute group. For example, the Item Supplier Country Location form has 4 preset keys. You can add only 14 additional attributes to this user interface and this can be a combination of character, number, and date type attributes.

When planning the attribute groups, in order to properly validate and display the information in the screens, you must determine the following in addition to the attributes themselves:

- **Display Order** - The order in which the attribute group set will appear in the Options menu of the relevant entity.
- **View Name** - The name that will be used for the database view that will be created for this group. Similar to the view defined at the group set level, this view will contain all attributes in the group. A view is a required information for each group and is used to facilitate querying attribute information for the group.
- **Validation Function** - Validates the information entered before the users save and close the CFAS screen. A validation function is not mandatory, but may be needed when further validation is required beyond that set at the attribute level. In such a case, you will need to write a custom package function for the validation function based on your business need.
- **Labels** - Sets the business name for the group and will be the name that appears on the screen. You must set at least one label for the default language for each group set you create. You can choose to create labels for more (alternate) languages based on your business need.

Attributes

Attributes are the bottom layer of the CFAS hierarchy. As mentioned above, you can have only 22 attributes per attribute group. Of those 22 attributes, only 10 are allowed to be character based attributes, 10 are number based attributes, and 2 are date attributes. You must consider this set limit when planning the attributes to be included in each group. Additionally, you must also determine and consider the following information when planning the attributes:

- **Data Type** - Indicates the type of data for the attribute. You can set this as a Number, Varchar, or Date.
- **Widget Type** - Indicates the type of the field that will appear for the attribute. You can select one of the following options:
  - Text Item - You can use this type for both Number and Varchar data types. When used, the attributes field will appear as a text box on the screen.

  - List of Values (LOV) - You can use this type for the Varchar data types only. A list of values appears as a text box and an LOV button. If you choose to use this widget type, you must also specify a record group. For more information, see Record Groups.
  - List Item - You can use this type for the Varchar data type only. When used, the attributes field will appear as a drop-down list box on the screen. If you choose to use this widget type, you must also specify a code type. For more information, see Codes.
  - Check box - You can use this type for Varchar data type only. When used, the attributes field will appear as a check box on the screen.

- **Display Sequence** - Indicates the order in which the attributes will appear in the attribute screen, from top to bottom. Attributes will appear in a single column on the screen.
- **Enabled** - Indicates whether the attribute appear as enabled or disabled (greyed out); only enabled attributes can be updated in the attribute screen. For attributes where you want the users to enter information, you must set them as enabled. For attributes that will display a default value (using a default function) that cannot be changed, you will need to set them as disabled.
- **Validation Function** - Validates the information for the attribute at the field level. A validation function is not mandatory, but may be needed when further validation is required beyond that set at the attribute level. In such a case, you will need to write a custom package function for the validation function based on your business need.
- **Maximum Length -** Indicates the maximum number of digits or characters that are allowed in the attribute, as well as the width of the attribute on the screen. You must specify this information for the attributes with Number or Varchar data types. This may not apply when you choose a List Item or Check box as the widget type. The maximum length for the List Item widget type is automatically set to 6 and the maximum length for the Check box widget type is automatically set to 1. For attributes with Number data type, you must enter the full length of the number. This includes the length to be allowed after the decimal point and positive/negative sign (if negative numbers are allowed). For example, if a particular attribute needs to allow for five digits with up to two digits after the decimal point and allow negative integers, you will need to specify the length as 9 (1 character for positive/negative sign, 5 digits before the decimal point, 1 decimal point, and 2 digits after the decimal point).

  Note: If you choose a record group, the maximum length will be automatically selected based on the value in the first column of the record group query.

- **Minimum/Maximum Value Allowed -** Applies to attributes with Number or Date data types. Indicates the minimum and maximum numeric or date value that can be entered for the attribute. Although this is not mandatory, you must consider this as part of the planning process.
- **Required** - Indicates whether the attribute will be considered as a mandatory field and trigger the standard CFAS validation. Once an attribute is activated, it is recommended that you avoid changing this information.

Other Setup Data

As part of planning the attributes, you must also review and set up the following as needed:

- Record Groups
- Codes

Record Groups

Record groups are needed when you select the List of Values widget type for the attributes. Record groups need to be set up ahead of time and it requires you to be familiar with writing PL/SQL queries. Each record group must contain just two data elements—an ID and a description, in this specific order (for example, supplier and supplier name). The RMS application already includes set record groups for the attributes associated with the pre-enabled entities. But, you may need to create additional record groups to support the attributes you add, based on your business need. The CFAS maintenance screen for the record groups (Record Groups Maintenance screen) enables you to create the record groups and build simple or complex queries for each record group. Although you can use this screen to write and test the queries, there is no preview feature available for the List of Values (LOV).

Codes

Codes are needed when you select the List Item widget type for the attributes. Codes for CFAS are similar to the codes that are used in other parts of RMS. They are made available separately for the CFAS framework for usability purposes. In case you want to use the codes that are part of the main RMS Codes tables, you can choose to copy over these tables using scripts. To specify separate codes for CFAS manually, use the CFAS maintenance screen for the codes (Codes Maintenance screen). This screen enables you to create the code header and detail records to support the list items for CFAS.

Custom Functions

There are three types of custom functions that CFAS is designed to support. For all of these function types, you will need to write a custom package function based on the CFAS standard in order to perform the functions described below.

Custom functions are not mandatory, but may be required when the custom attributes need anything other than the standard validation that is done based on the attribute setup (for example, ensuring that required attributes are entered).

Qualifier Function

Qualifier Functions are at the attribute group set level only and are used to ensure that all the required information is entered or available before the users open the CFAS screen from the Options menu. The base validation ensures that the basic information (usually the primary key) for the entity is entered before launching the CFAS screens. In case you want to ensure such a validation based on any other information, you can create a qualifier function. Qualifier functions are not mandatory.

For example, in the Ordering dialog, you may use a qualifier function to ensure that at least one item is added to the order before accessing one of the CFAS screens from the Order Header screen.

Default Function

Default Functions are also only at the attribute group set level. These types of functions are used to set the default values for the attributes in the attribute group set based on some previously determined criteria. For example, copying information from the subclass level to the item level when the same attributes are set up for both the entities.

Default functions are not mandatory. Once implemented, the default function will run each time the users open the relevant screen. You must ensure that the custom function is written in such a manner that it only sets the default values when attributes are set up for the first time in an entity. For example, when the user changes the value to something other than the default, the value will not be overwritten the next time the screen is opened.

Validation Function

Validation functions can be defined at the entity, group set, group, and attribute level. Validation functions are not mandatory. These functions enable you to add additional validation into the attribute screens above the default validation provided with the base functionality. The default validation in the screens is based on how the attributes are created (for example, required attributes are entered; dates do not exceed the maximum allowable value, and so on).

For more information on the existing validation functions in the CFAS user interface and setting up your own custom validation functions, see CFAS User Interface Validation Routines.

Since the validation functions work slightly different based on the level at which it is set, determining the level for the validation is an important part of the planning process. Validation at the attribute group set level is executed when users click the OK button on the attribute group set screen. Validation at the attribute group level is executed when users switch tabs between attribute groups in the attribute group set screen. It will also be executed for the attribute group currently being displayed when users click the OK button. Validation at the attribute level is executed when users navigate to other attribute fields by pressing the Tab key.

The following examples illustrate some validation scenarios that will help you determine the validation level based on your business need:

- Entity - Entity level validation is the highest level and triggered by the pre-enabled base RMS forms usually via the OK button in the form (to check if there are missing required attributes).
- Attribute Group Set - If the validation requires the attributes to be added in more than one group to perform the validation, set the validation at this level. For example, consider a scenario where you have set up store attributes with one group set that includes the alternate address information for a store and another group set the indicates that the store has an offsite backroom. In case you want to check and ensure that the store with an offsite backroom also has the alternate address entered, you may need to set the validation at the attribute group set level.
- Attribute Group - Set the validation at this level, if you want the function to validate attributes within a single attribute group. For example, consider an attribute group that captures the address information. You may need to set the validation at the attribute group level to ensure that the users enter the State information when they select the country as USA.

- Attribute - Set the validation at this level to perform validations at the field level. For example, consider that you created an item level attribute called Related Item. To ensure that it is a valid item number in the system, you may need to set a relevant validation function at the attribute level.

Other Custom Functions

The custom functions described above are all related to setting up the attributes when a new entity is created or updated. They do not include the facility to use the attributes anywhere within the RMS application or external systems. In order to use the custom attributes for driving functionality within the RMS application, you may need to create additional custom functions. Such custom functions will need to leverage the views (to query data) that are created as part of the attribute setup process.

3

Planning Your Implementation

Before you proceed with implementing CFAS, you must meet certain pre-requisites, understand the process flows, and review certain CFAS-specific considerations. This chapter provides more information that will help you prepare for implementing or leveraging CFAS based on your business need. It includes the following topics:

Prerequisites

Before proceeding, ensure you meet the following pre-requisites:

Required Skills and Expertise

This document is intended for application integrators and implementation personnel. Implementing CFAS or leveraging the CFAS framework to set up additional attributes requires that you are familiar with or have access to the following:

- RMS functional and technical concepts, including knowledge of any existing implementation processes at the retailer.
- Oracle Forms-based development.
- PL/SQL and UNIX commands, including shell configuration and scripting.
- SQL and Database Administrator (DBA) level commands and tasks.

Required Development Kit/Software

To implement CFAS, you must also be familiar with and have access to the following development tools:

- Oracle Forms Services 10g Release 2
- SQL Editor, such as Oracle SQL Developer

Associated RMS (Retail Apps) Version

The CFAS framework is currently enabled and associated only with RMS F Release. To start using the CFAS framework, ensure that you have installed or upgraded to F Release.

CFAS Administration Role

Only a selected group of users will need to access the CFAS Maintenance users for setting up the CFAS entities and attributes. Ensure that you set up a CFAS Administrator (for example, CFAS_ADMIN) role. Only users with the CFAS_ADMIN role can view and access the CFAS Maintenance screens in the Custom Flex Attribute Setup folder.

The Custom Flex Attribute Setup folder will not appear for the users without the CFAS_ADMIN role. Once set up, the CFAS extension user interface will be accessible to all RMS users from the Options menu of the relevant RMS form. Users will be able to view the set extension attributes, capture, or store the relevant information in the CFAS user interface.

For more information on user management and assigning roles, refer to the *Oracle Retail Merchandising System User Guide*.

Implementation Plan

Before you proceed with the implementation, ensure that you are familiar with the CFAS concepts and have a specific implementation plan based on your business requirements. The implementation plan must at least (not limited to) address the following:

- Appropriate level and entities to be extended.
- Number of attribute group sets, groups, and attributes.
- Functions and scripts to be designed and used.
- Plans for maintenance and future growth or extensions.

Process/Business Flows

This section provides the following process flow that you can follow to plan the attributes you want to set up and make them available in the RMS production environment. It also describes the various objects involved in the planning process and provides key considerations:

- Phase 1 - Plan Entities and Attribute Group Sets
- Phase 2 - Plan Attribute Groups and Attributes
- Phase 3 - Set Up CFAS Objects Using the CFAS Maintenance Screens
- Phase 4 - Test and Activate the CFAS User Interface

Phase 1 - Plan Entities and Attribute Group Sets

In this phase:

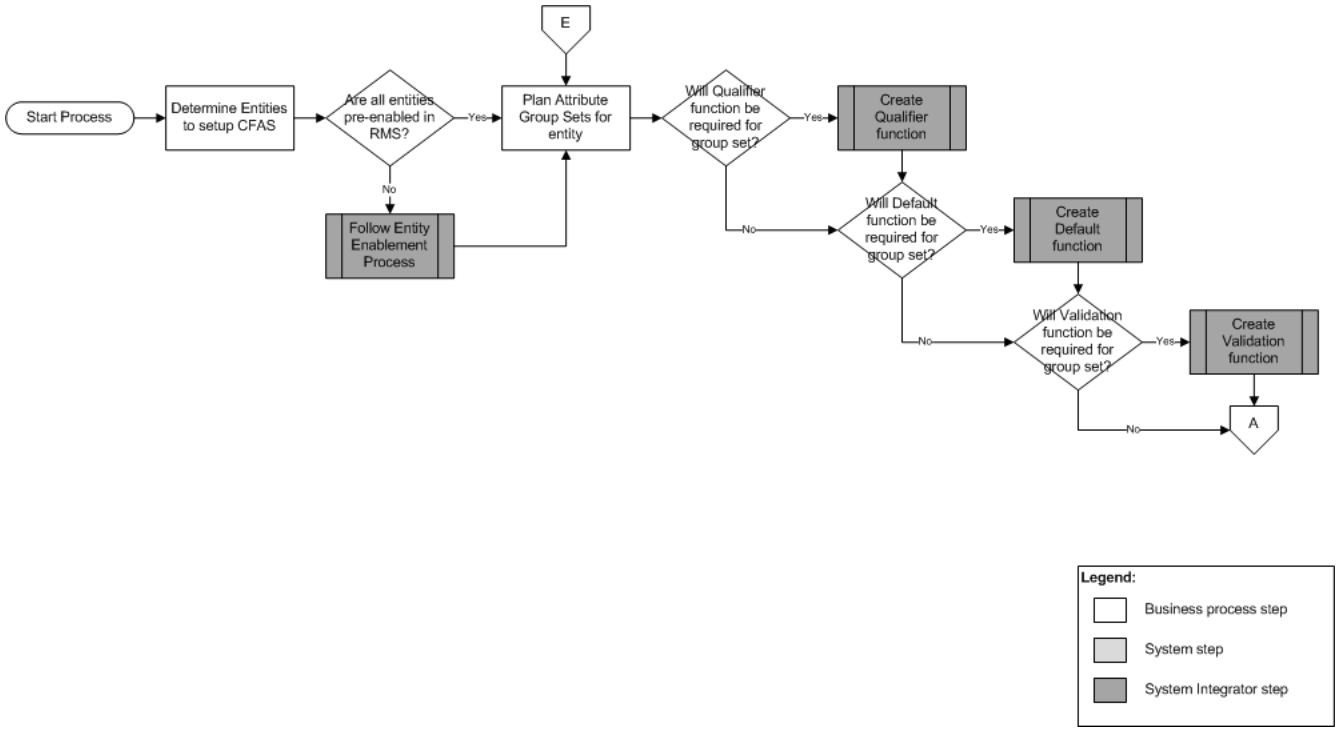
1. Determine the entities that need to be set up for CFAS. By default, RMS includes pre-enabled entities that you can leverage out of the box. For more information, see Entities.

   In case you want to set up an entity (other than the pre-enabled entities), refer to the My Oracle Support Note – *Oracle Retail Merchandising System Extending the Custom Flex Attribute Solution [ID 1285660.1]*.

2. Plan the attribute group sets for the relevant entity. Once set up and activated, the attribute group sets appear as the additional option in the Options menu for the relevant entity. As part of the attribute group sets planning, you may also need to determine and set the qualifier, default, and validation functions. For more information, see Attribute Group Sets.

3. Once you have planned the entities and attribute group sets, proceed to Phase 2.

Figure 3–1 Plan Entities and Attribute Group Sets

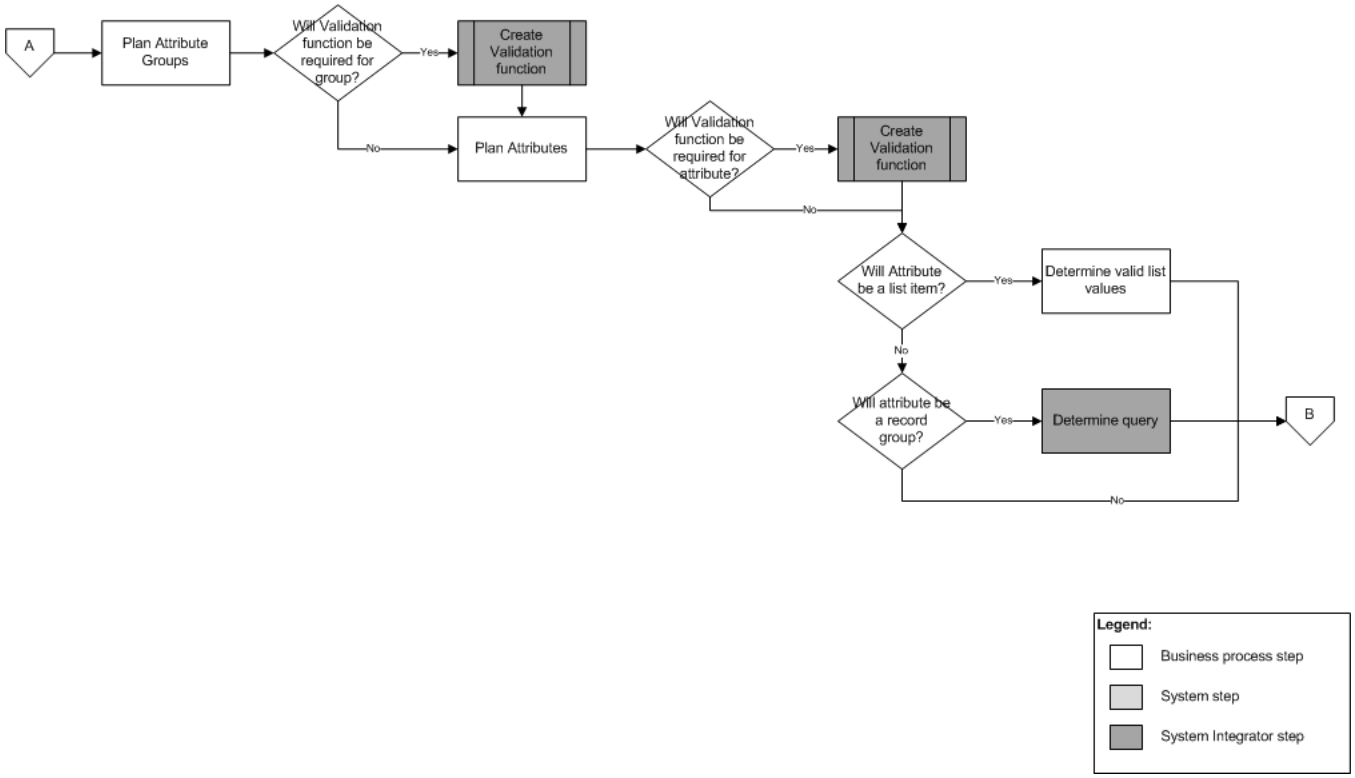


Phase 2 - Plan Attribute Groups and Attributes

In this phase:

1. Plan the attribute groups for the relevant planned attribute group sets along with any required validation function. For more information, see Attribute Groups.
2. Once you have planned the attribute groups, plan the attributes along with the data type, widget type, and any required validation function. For more information, see Attributes.

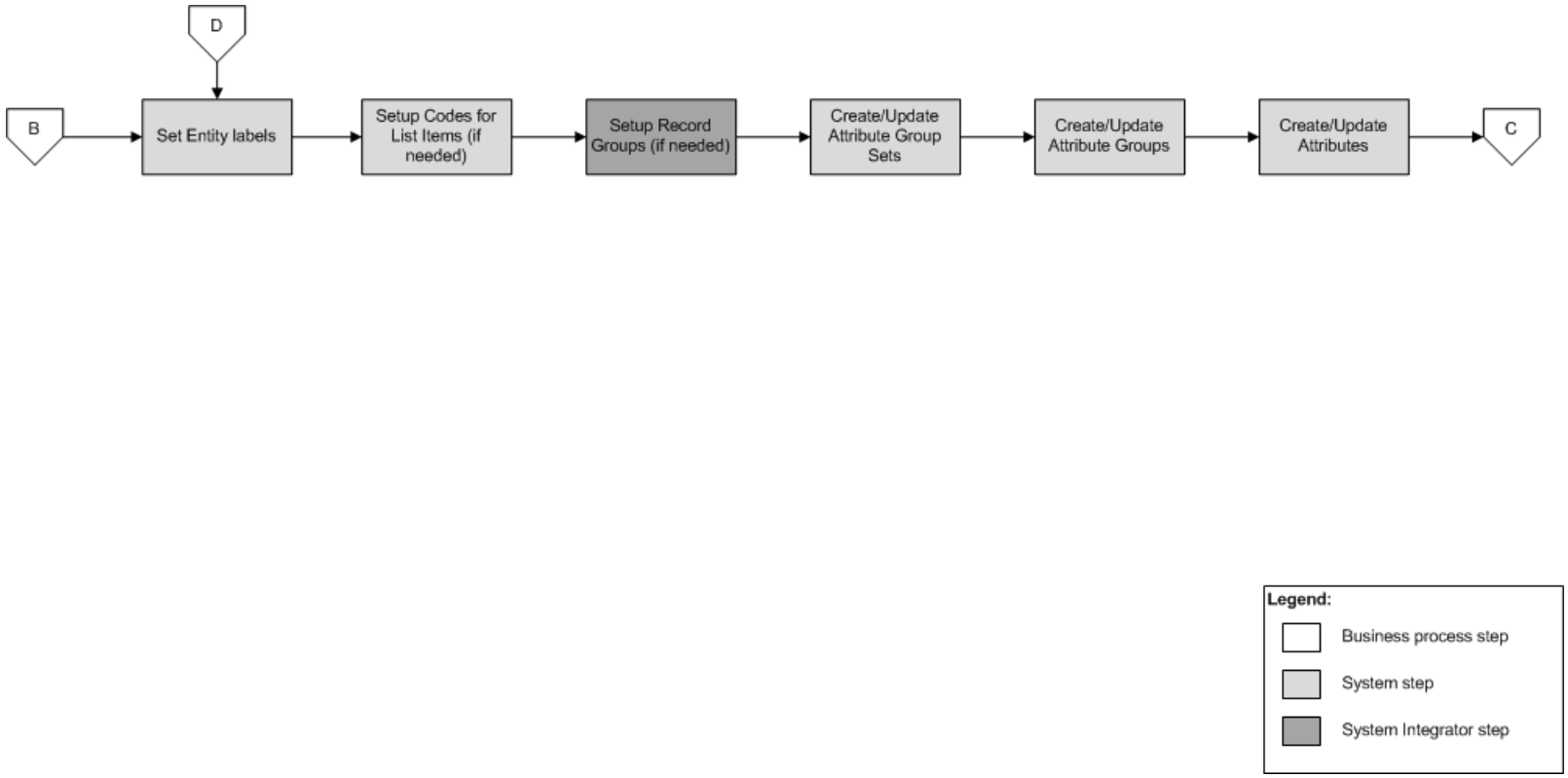
Figure 3–2 Plan Attribute Groups and Attributes



Phase 3 - Set Up CFAS Objects Using the CFAS Maintenance Screens

In this phase, use the CFAS Maintenance Screens to set up entity labels, codes, record groups, attribute group sets, attribute groups, and attributes. For more information, see CFAS Maintenance Screens.

Figure 3–3 Set Up CFAS Objects Using the CFAS Maintenance Screens

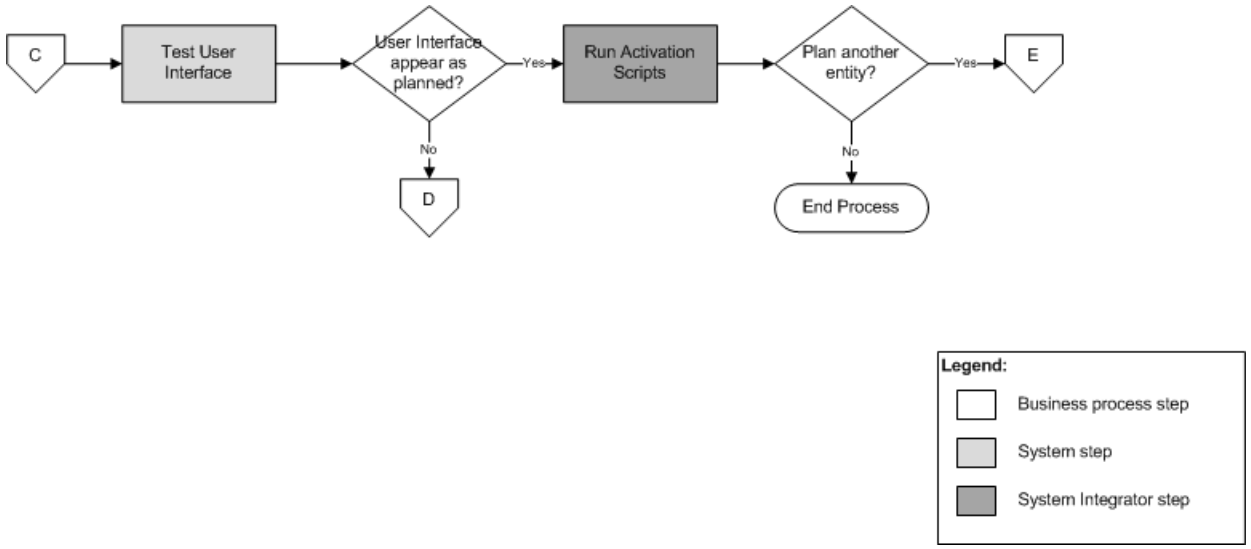


Phase 4 - Test and Activate the CFAS User Interface

In this phase:

1. Test the user interface. For more information, see Setting Up Attributes.
2. Once it is confirmed that the attribute screens have been set up as planned, run the cfagen.ksh batch script to create the database objects, including the views and staging tables. After the script has been run, the attribute group sets will be visible to all users in the appropriate RMS screen menus. Additionally, the setup screens in RMS will reflect that the various levels are considered active. For more information, see CFAS Database Scripts.

Figure 3–4 Test and Activate the CFAS User Interface



Integration Considerations

This section describes the two primary ways that other systems use to integrate with RMS and their relevance with the CFAS framework. It includes the following:

- Integration with Retail Integration Bus (RIB)
- Flat File Integration

Integration with Retail Integration Bus (RIB)

In this release of CFAS, RIB integration APIs are not pre-enabled. If the existing RIB integration points between Oracle Retail (for example, Store Inventory Management or Warehouse Management System) or other third party applications require these attributes to be included, additional customization will be required to enable this integration. It is recommended that the custom node in RIB be used for these changes.

Flat File Integration

Flat file integration is supported in CFAS. Staging tables are generated for each new entity that is set up. Load scripts are also generated that move your staging data into CFAS extension tables for continued use. Examples of these data sources are legacy integration feeds, one-time conversions, or third party integration.

Internationalization Considerations

In CFAS, the following database tables are used to store the relevant labels in different languages:

- CFA_EXT_ENTITY_KEY_LABELS – stores entity key labels.
- CFA_ATTRIB_GROUP_SET_LABELS – stores attribute group set labels.
- CFA_ATTRIB_GROUP_LABELS – stores attribute group labels.
- CFA_ATTRIB_LABELS – stores attribute labels.
- CFA_REC_GROUP_LABELS – stores record group labels.
- CFA_CODE_DETAIL_DESCS – stores code labels.

For more information on the database tables mentioned above, see CFAS Table Definitions.

For more information on the supported languages, refer to the *Oracle Retail Merchandising System Operations Guide, Volume 3 - Back End Configuration and Operations*.

Additional Considerations

Before proceeding with implementing CFAS, review the following considerations:

- The framework is enabled only on select RMS features. You can use the framework and patterns to enable customization for other entities. However, such customizations are not supported by Oracle.
- The architecture is based on the localization framework and does not adhere to the Oracle Fusion customization framework. As such, upgrade strategy to the Oracle Fusion framework is not in scope of this document.
- Custom defined fields are currently not exposed to other applications, and not in scope of this document. An additional effort will be involved to expose such fields to downstream applications using RIB, exports, or other means.
- CFAS attributes are scalar. This means that only one value of that attribute can be stored per entity. To set up multi-record attributes, you can use groups as multi-records.
- To avoid any data security issues, you must apply relevant security policies on the extension tables and access views.

- RMS Entities are extended at the database table level. For example, items may be extended at the header level - ITEM_MASTER (in the ITEM_MASTER_CFA_EXT table), item supplier level - ITEM_SUPPLIER (in the ITEM_SUPPLIER_CFA_EXT table), item location level - ITEM_LOC (in the ITEM_LOC_CFA_EXT table), and so on. Maintenance screens for these tables will have separate individual access to the relevant extensions.
- When an entity record is deleted (order), the related CFAS attributes associated with that order will also be deleted for all entities that are pre-enabled as part of the base functionality.
- Performance within the CFAS solution will be relative to the size of the entity you are extending. There is a 1 to 1 ratio of CFAS records to entity records (that is extending the Item/Location table will create a table with an equivalent number of records). To ensure that you can maintain the performance of the entity in a production environment, it is recommended that you first perform proof-of-concepts on the entities.
- If you plan to enable entities other than those that are pre-enabled as part of the base installation, it is recommended that you use lower volume transactions, such as the header of an inventory transaction, in place of the detail record.

4

CFAS Maintenance Screens

Once the attribute structure and other relevant details have been planned out, you can set the attributes up in the system using the CFAS Maintenance screens. The CFAS Maintenance screens are set of screens that appear under the Custom Flex Attribute Setup group in the RMS Start Menu. This chapter describes how you can use the CFAS Maintenance screens to set up the CFAS attributes. It includes the following sections:

Additional Considerations

The following considerations apply to all the screens mentioned in this chapter:

CFAS Administration Role

To access the Custom Flex Attribute Setup group and CFAS Maintenance screens, ensure that you have the CFAS Administrator (CFAS_ADMIN) role associated with your user account. For more information, see CFAS Administration Role.

Character Limit for Labels

The CFAS Maintenance screens enable you to set up labels for the attribute, attribute groups, attribute group sets, record groups, codes, and entities. When activated, the labels set for the attribute group sets are the screen text that appear on the Options menu and labels set for other objects are the field prompts in the CFAS UI. To avoid any overlap issues in the user interface and data consistency, it is recommended that you maintain a 60 character limit for all the labels and 40 character limit for the entity key labels.

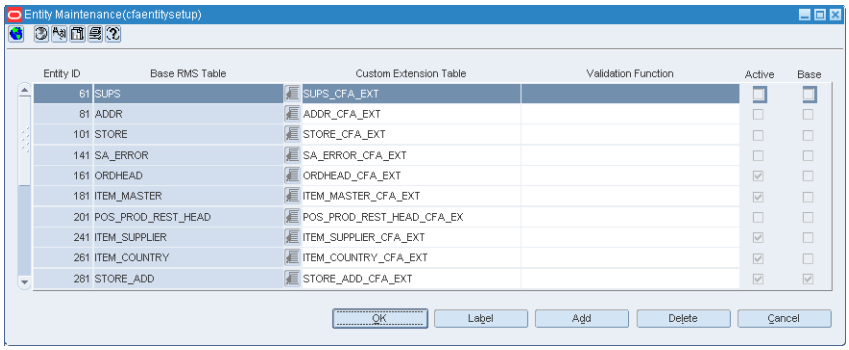
Setting Up Entities and Entity Labels

The Entity Maintenance screen enables you to review and set up entities and entity labels for your CFAS-based implementation. For more information on entities, see Entities.

To set up entity labels:

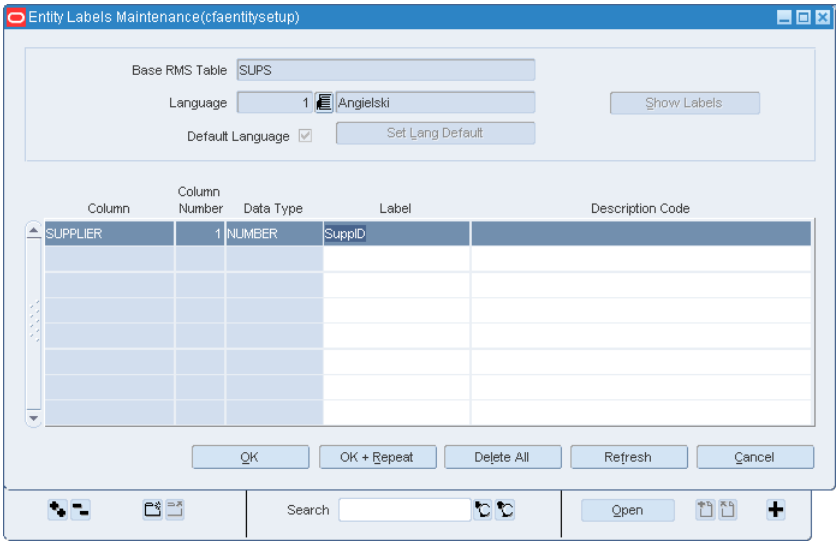
1. From the main menu, expand **Custom Flex Attributes Setup**, and then click **Entity Setup**.
2. From the **Contents of Entity Setup** area, double click **Edit**. The **Entity Maintenance** screen appears.

Figure 4–1 Entity Maintenance Screen



3. On the **Entity Maintenance** screen, select the base RMS table corresponding to the entity you want. For more information, see Adding an Entity.
4. Click **Label**.The **Entity Labels Maintenance** screen appears.

Figure 4–2 Entity Labels Maintenance Screen



5. On the **Entity Labels Maintenance** screen, select the relevant language from the **Language** list of values, and then click **Show Labels**.
6. Under the **Labels** column, enter the labels for the relevant columns, enter a relevant function name under **Description Code**. Description Code column must

contain the function that will display the description of the relevant key. For example, SUPPS_SQL.GET_SUPP_NAME will retrieve the supplier name.

7. In case you want to set the selected language as the default, click **Set Lang Default**. The **Default Language** check box appears selected.
8. Click **OK+Repeat** to set labels for another language.
9. Once the labels are set up, click **OK** to close the **Entity Labels Maintenance** screen.
10. Click **OK** to close the **Entity Maintenance** screen.

Adding an Entity

To add an entity that is not currently extended, click Add. A new line entry appears on screen.

1. Under **Base RMS Table**, enter the relevant base RMS table name using the LOV button. The **List of Base RMS tables** window appears.
2. Find and select the table that is not currently extended, and click **OK**. Notice that the base RMS table name gets added under the Base RMS Table column and a new corresponding extension table name (with the _CFA_EXT suffix) gets added under the Custom Extension Table column.
3. Optional. Under the **Validation Function** column, you can choose to specify any validation function you have planned at the entity level.
4. Click **Label** and follow steps 4 through 10 to set up labels.

Note: On the Entity Maintenance screen, the Active check box indicates that the CFAS attribute is active for the relevant entity. It also means that the relevant database tables and views are created and ready for use. The Base check box indicates that the relevant entity is pre-enabled by default in RMS.

Setting Up Codes

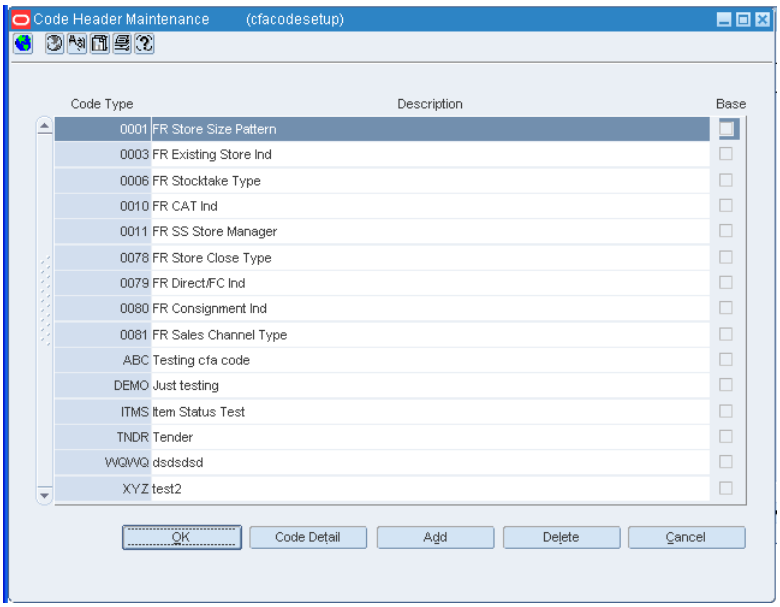
The Code Header Maintenance screen enables you to set up codes that will be used as values in the attributes with the List Item widget type. For more information on codes, see Codes.

To set up codes:

1. From the main menu, expand **Custom Flex Attributes Setup**, and then click **Code Setup**.

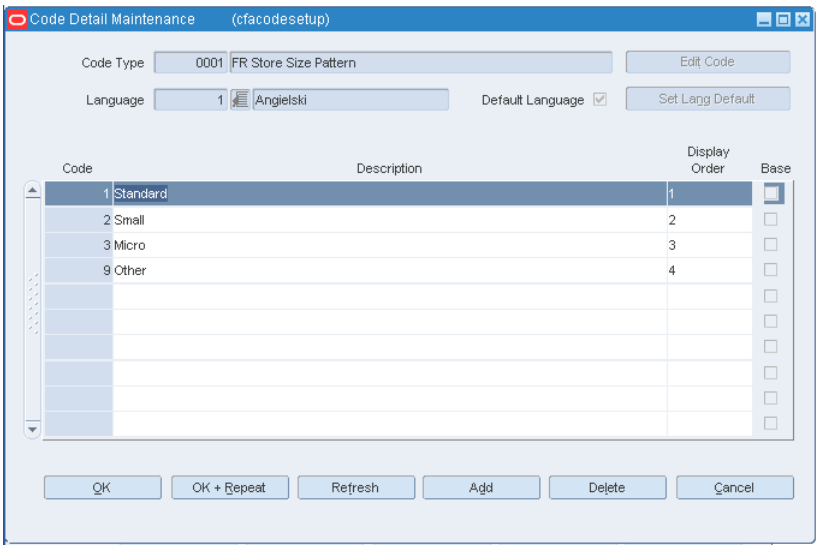
2. From the **Contents of Code Setup** area, double click **Edit**. The **Code Header Maintenance** screen appears.

Figure 4–3 Code Header Maintenance Screen



3. On the **Code Header Maintenance** screen, click **Add**. A new line entry appears on screen.
4. Enter a valid code type under **Code Type**, and add a relevant description.
5. Keeping the new code type selected, click **Code Detail**. The **Code Detail Maintenance** screen appears.

Figure 4–4 Code Detail Maintenance Screen



6. In the **Code Detail Maintenance** screen, select the relevant language using the **Language** LOV button, and then click **Edit Code**.
7. Enter relevant code, associated descriptions, and set a display order.

8. In case you want to set the current language as the default, click **Set Lang Default**.
9. Click **OK+Repeat** to set codes in other languages.
10. Click **OK** to apply and close the **Code Detail Maintenance** screen.
11. Click **OK** to close the **Code Header Maintenance** screen.

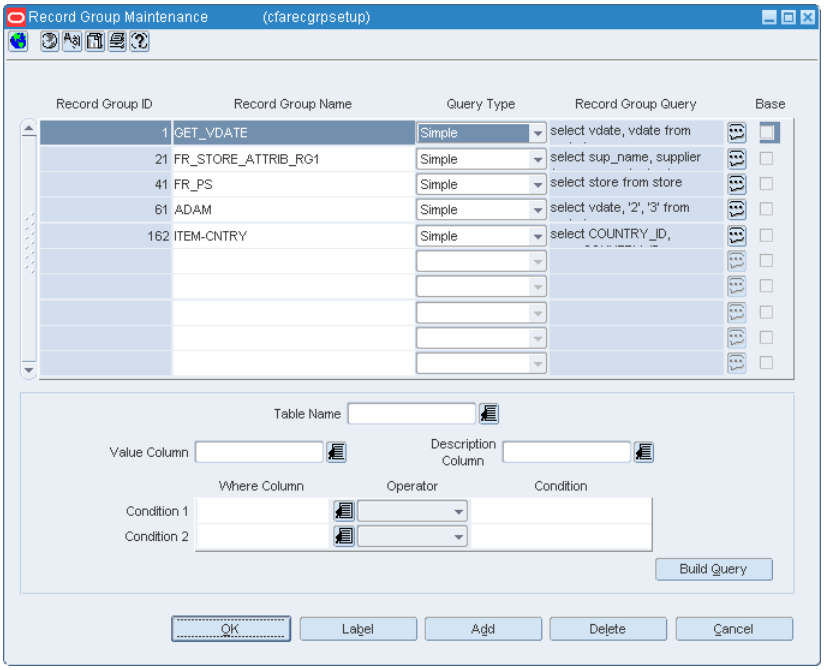
Setting Up Record Groups

The Record Group Maintenance screen enables you to set up record groups for the attributes with List of Values (LOV) widget type. For more information on record groups, see Record Groups.

To set up record groups:

1. From the main menu, expand **Custom Flex Attributes Setup**, and then click **Record Group Setup**.
2. From the **Contents of Record Group Setup** area, double click **Edit**. The **Record Group Maintenance** screen appears.

Figure 4–5 Record Group Maintenance Screen



3. On the **Record Group Maintenance** screen, click **Add**. A new line entry appears on screen.
4. Under **Record Group Name**, enter a valid name for the record group.
5. Under **Query Type**, select one of the following query types:
   - For simple queries:
     a. Under **Query Type**, select **Simple**. The **Table Name** field gets enabled.
     b. Use the **Table LOV** button and select the relevant database table for the query. Once you select the table, the other fields get enabled.

c. Select the relevant column and description column for the query in the **Value Column** and **Description Column** fields.

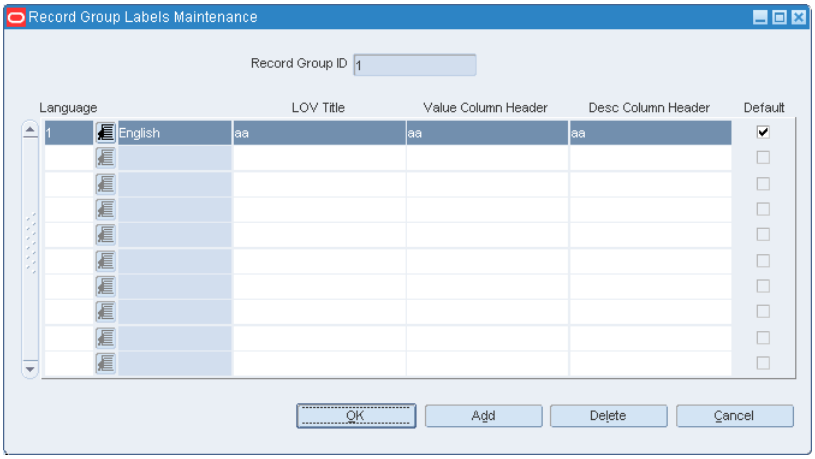
d. Set a condition for the query by specifying values under **Where Column**, **Operator**, and **Condition** columns.

e. Click **Build Query**. Notice that the record group query gets added under the **Record Group Query** column.

- For complex queries:
  - Under **Query Type**, select **Complex**. Unlike simple queries, the user interface does not provide you the ability to set a query. You must work with your database administrator to build and record the query for your record group.

6. Once the query is set up, click **Label**. The **Record Group Labels Maintenance** screen appears.

Figure 4–6 Record Group Labels Maintenance Screen



7. On the **Record Group Labels Maintenance** screen, select the language using the **Language** LOV button.
8. Under **LOV Title**, set a relevant list of values title.
9. Under **Value Column Header** and **Desc Column Header**, set a header name for the value column and description column.
10. Add more lines to enter similar information for other languages.
11. Under **Default**, select the check box next to the language you want to set as the default language.
12. Click **OK** to apply and close the Record Group Labels Maintenance screen.
13. Click **OK** to close the Record Group Maintenance screen.

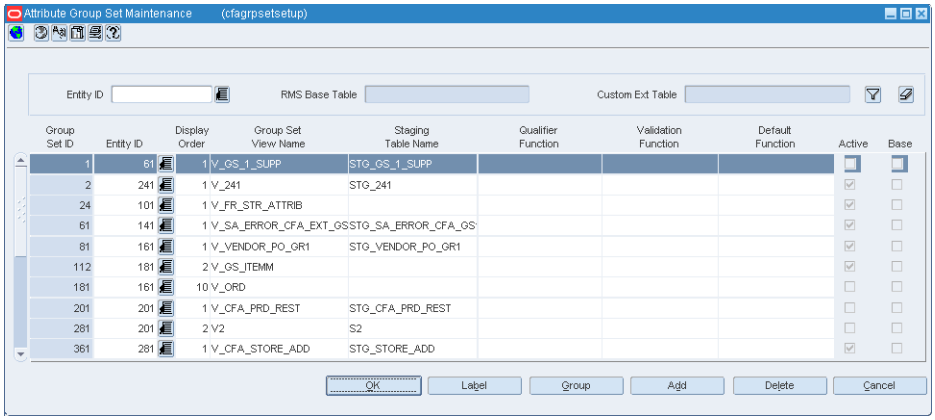
Setting Up Attribute Group Sets

The Attribute Group Set Maintenance screen enables you to set up attribute group sets. For more information on attribute group sets, see Attribute Group Sets.

To set up attribute group sets:

1. From the main menu, expand **Custom Flex Attributes Setup**, and then click **Group Set Setup**.
2. From the **Contents of Group Set Setup** area, double click **Edit**. The **Attribute Group Set Maintenance** screen appears.

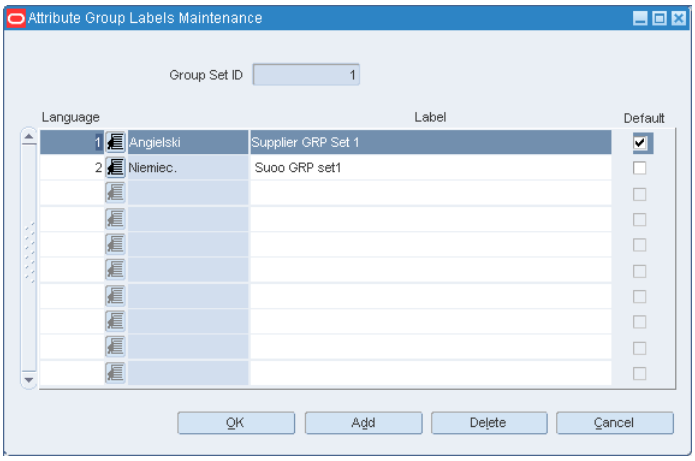
Figure 4–7 Attribute Group Set Maintenance Screen



3. On the **Attribute Group Set Maintenance** screen, click **Add** to add a new attribute group set. A new line entry appears on screen.
4. Enter relevant information in the following columns:
   a. Under **Entity ID**, enter the relevant Entity ID using the list of values button. This will be the identification code of the entity you set up in the Entity Maintenance screen.
   b. Under **Display Order**, enter a valid number that indicates the order in which the attribute group set label will appear in the Options menu of the relevant RMS base form.
   c. Under **Group Set View Name**, enter a valid group set view name for the attribute group set. It is recommended that you start the name with V_.
   d. Under **Staging Table Name**, enter a staging table name that you want to create and associate with this attribute group set. It is recommended that you start the name with STG_.
   e. Under **Qualifier Function**, **Validation Function**, and **Default Function**, enter the planned functions for the attribute group set. For more information, see Attribute Group Sets.

5. Click **Label**. The **Attribute Group Labels Maintenance** screen appears.

Figure 4–8 Attribute Group Labels Maintenance Screen



6. On the **Attribute Group Labels Maintenance** screen, under **Language**, select the relevant language code using the list of values button.
7. Under **Label**, enter the name that will appear for the attribute group set in the Options menu of the base RMS form.

   You can define a hot key by putting a **&** character before any character in the label you want. This makes the attribute group set label in the Options menu to appear with a hot key. It can be used as a keyboard shortcut to quickly access the CFAS user interface.

   For example, consider that you want to set up a attribute group set label as **Create Special Order**. To set up the letter **S** as the hot key, enter the **&** character before the letter **S** in the **Attribute Group Labels Maintenance** screen. The label will then read as **Create &Special Order**. Once activated, the label name in the Options menu will read as **Create Special Order**. Notice the underline under the letter **S** which indicates that it is the designated hot key for the menu option. Ensure that you maintain a list of assigned hot keys to ensure that they remain unique for each base RMS form.
8. You can choose to set up the labels for more languages. In case you set up more languages, ensure that you select the relevant check box under **Default** for the default language.
9. Once done, click **OK** to close the Attribute Group Labels Maintenance screen.
10. Once all attribute group sets are set up, click **OK** to close the Attribute Group Set Maintenance screen.

You must now set up attribute groups. To open the **Attribute Group Maintenance** screen, you can access the screen from the main menu or click the **Group** button on the **Attribute Group Set Maintenance** screen.

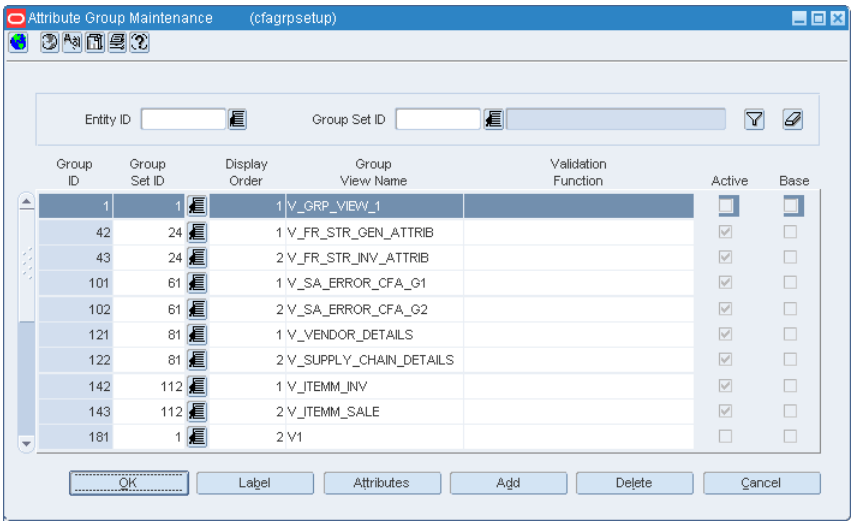
Setting Up Attribute Groups

The Attribute Group Maintenance screen enables you to set up attribute groups. For more information on attribute groups, see Attribute Groups.

To set up attribute groups:

1. From the main menu, expand **Custom Flex Attributes Setup**, and then click **Group Setup**.
2. From the **Contents of Group Setup** area, double click **Edit**. The **Attribute Group Maintenance** screen appears.

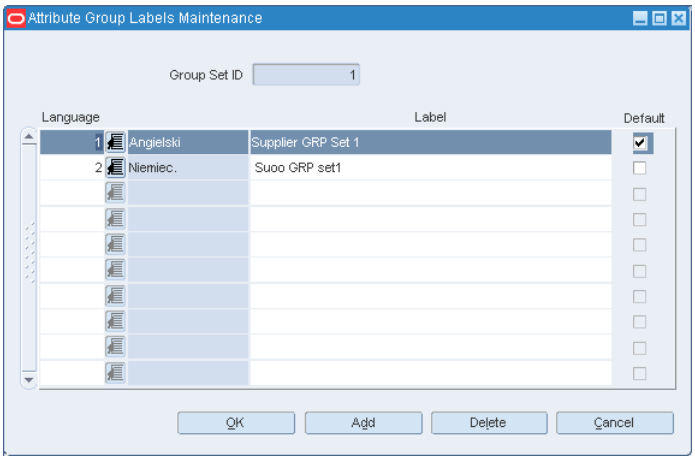
Figure 4–9 Attribute Group Maintenance Screen



3. On the **Attribute Group Maintenance** screen, click **Add** to add a new attribute group. A new line entry appears on screen.
4. Enter the following information in the columns:
   a. Under **Group Set ID**, enter the relevant group set ID using the list of values button. This will be the identification code of the attribute group set you set up in the Attribute Group Set Maintenance screen.
   b. Under **Display Order**, enter a valid number that indicates the order in which the attribute group label will appear.
   c. Under **Group View Name**, enter a valid group view name for the attribute group. It is recommended that you start the name with V_.
   d. Under **Validation Function**, enter the planned function for the attribute group set. For more information, see Attribute Groups.

5. Click **Label**. The **Attribute Group Labels Maintenance** screen appears.

Figure 4–10 Attribute Group Labels Maintenance Screen



6. On the **Attribute Group Labels Maintenance** screen, under **Language**, select the relevant language code using the list of values button.
7. Under **Label**, enter the name that will appear for the attribute group set in the Options menu of the base RMS form.
8. You can choose to set up the labels for more languages. In case you set up more languages, ensure that you select the relevant check box under Default for the default language.
9. Once done, click **OK** to close the Attribute Group Labels Maintenance screen.
10. Once all attribute group sets are set up, click **OK** to close the Attribute Group Maintenance screen.

You must now set up attributes. To open the **Attribute Maintenance** screen, you can access the screen from the main menu or click the **Attributes** button on the **Attribute Group Maintenance** screen.

Setting Up Attributes

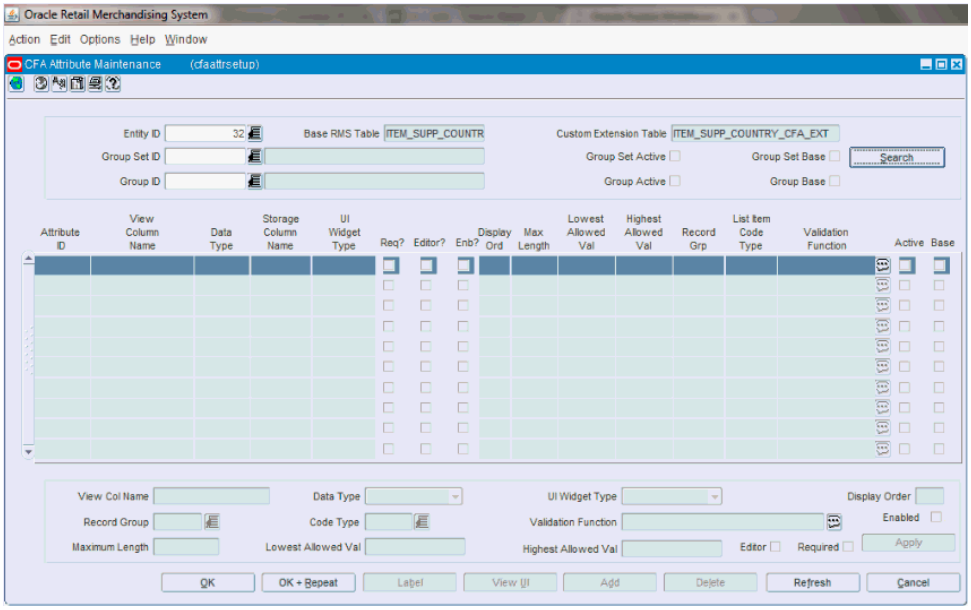
The Attribute Maintenance screen enables you to set up attributes. For more information on attributes, see Attributes.

To set up attributes:

1. From the main menu, expand **Custom Flex Attributes Setup**, and then click **Attribute Setup**.

2. From the **Contents of Attribute Setup** area, double click **Edit**. The **Attribute Maintenance** screen appears.

Figure 4–11 Attribute Maintenance Screen



3. On the **Attribute Maintenance** screen, select the relevant entity ID, attribute group set ID, and group ID set up in the Entity Maintenance, Attribute Group Set Maintenance, and Attribute Group Maintenance screens.
4. Click **Add** to add a new attribute.
5. Enter information in the following fields:
   a. In the **View Col Name** field, enter a valid column name.
   b. In the **Data Type** drop-down list, select a relevant data type. You can choose from VARCHAR2, NUMBER, and DATE.
   c. In case you selected VARCHAR2 or NUMBER in the **Data Type** field, the **UI Widget Type** drop-down list gets enabled.
   d. In the **UI Widget Type** drop-down list, select a relevant widget type. Based on the data type you selected, the following options appear in the **UI Widget Type** drop-down list:

      – For VARCHAR2 data type, the Text Item, List of Values, List Item, and Check Box widget types appear.

      – For NUMBER data type, the Text Item and List of Values widget types appear.
   e. Set a display order in the **Display Order** field.
   f. In case you selected **List of Values** widget type, the **Record Group** field gets enabled. Use the LOV button to select one of the available record groups. Record groups are set up in the Record Groups Maintenance screen. For more information, see Setting Up Record Groups.

g. In case you selected **List Item** widget type, the **Code Type** field gets enabled. Use the LOV button to select one of the available code types. Code types are set up in the Code Type Maintenance screen. For more information, see Setting Up Codes.

Note: A current limitation in Oracle Forms causes the check box label to offset when selected. This may occur when you set up a single attribute with the Check box widget type.

To avoid this issue, it is recommended that you add another attribute to the screen.

---

h. In the **Validation Function** field, enter a valid validation function planned for the attribute.

i. In case you want to keep this attribute enabled in the CFAS user interface, keep the **Enabled** check box selected.

j. In case you selected the Text Item widget type, the **Max Length** field gets enabled. Set the maximum number of characters allowed in the field.

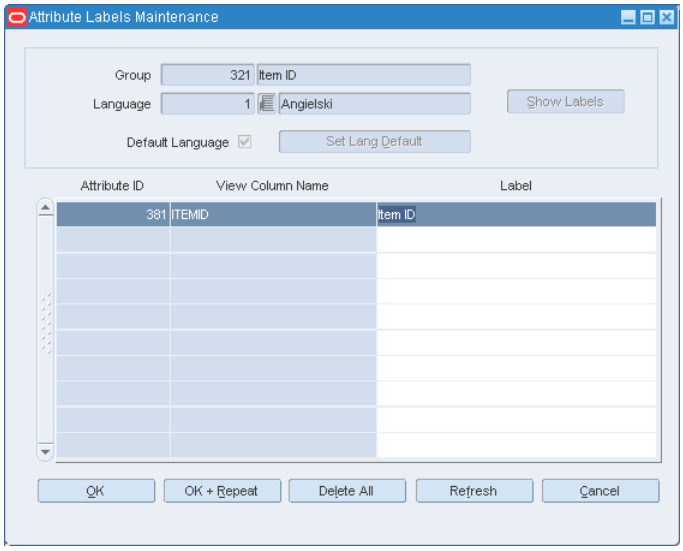
k. In case you selected the Number data type, the **Lowest Allowed Val** and **Highest Allowed Val** fields get enabled. Set the lowest and highest allowed values in the field.

l. In case you want to set this field as mandatory, select the **Required** check box. You must also ensure that the **Enabled** check box is selected.

6. Click **Apply**. Notice that a new attribute is added.

7. Once the attributes are added, click **Label**. The **Attribute Labels Maintenance** screen appears.

Figure 4–12 Attribute Labels Maintenance Screen



8. In the **Attribute Labels Maintenance** screen, select the relevant language using the **Language** LOV button, and then click **Show Labels**.

9. Under **Label**, enter relevant labels for the attributes. Click **OK+Repeat** to set up labels for other languages.

10. Click **OK** to close the Attribute Labels Maintenance screen.

Once you have set up all the information into the system, it is recommended that you test and review all the attribute group sets for each entity using the **View UI** button and **View CFAS UI Parameters** menu option in the Attribute Maintenance screen.

After verifying that all the screen layouts look correct and work as expected, you can proceed with running the cfagen.ksh batch script and make the CFAS attributes available for use in RMS. For more information on the database scripts, see CFAS Database Scripts.

5

Ongoing Maintenance

Once you set up the attributes and run the scripts to create the database entities, the flex attributes are available for viewing by all users in the system. Once these scripts are run, an indicator appears in the CFAS Maintenance screens displaying the group sets, groups, or attributes that have been activated.

This chapter describes the considerations for maintaining and upgrading the flex attributes that are in use. It includes the following sections:

Modifying Attributes

Once the flex attributes are activated, the changes that you can make at the group set, group, and attribute level are minimal. After activation, only the following can be updated using the CFAS Maintenance screens:

- **Label** - You can change the labels set at the group sets, groups, and attributes level at any time.
- **Display Order** - You can change the display order in which the group sets, groups, and attributes appear in the menus or attributes screens after the attributes are activated.
- **Validation Functions** - You can change the qualifier, validation and default function names at any time, but the functions will only apply to the changes made next time a user adds new attributes to an entity or updates existing attributes. Also, changes to the code inside these functions can be updated at anytime.

Additionally, you can continue to add new attributes, groups, and group sets even after an entity has had others activated. You must follow the same process of planning the attributes, adding them in the CFAS Maintenance screens, and then the last step being to run the database activation scripts.

Note: Although changes can be made directly in the database, it is not recommended.

For reasons that require modifying an active attributes' properties or deleting the attribute, a suggested approach is outlined below:

1. Backup the data in backup tables using the easy access view at the group set level for each group set defined for the entity.
2. Using a tool like SQL Developer or SQL plus, go to the CFAS metadata table and set the attribute's ACTIVE_IND to 'N' and commit.
3. Ensure there is a staging table defined at the group set level. If none set the group_set's active indicator to 'N' where the attribute is linked to.
4. Go to the setup up screen and update the attribute's properties - change the data type, widget type etc. or click the delete button (if deleting the metadata).
5. When the metadata updates are done, run the cfagen.ksh script to update the objects. Make sure the view and staging tables are created properly with the updated attribute.
6. Truncate the CFA custom extension table linked to the base table.
7. Using the backup tables you created earlier, load the staging table with the corrected data. This needs to be done per group set.
8. Run the CFA load script for each group set/staging table to load the data back to the extension table.

Deleting Attributes

Once the attributes have been activated, they cannot be deleted at any level using the CFAS Maintenance screens. This includes entity, group set, group, and attributes.

Although you can delete attributes directly in the database, it is not recommended. To delete attributes in the database, it is recommended that you follow instructions in this guide to make the necessary changes.

Note: You must delete attributes in a hierarchical manner. Entities cannot be deleted first. You must first start with attributes, and then proceed up the levels to entity.

Security Considerations

The CFAS screens use the same form level security as that exists in other parts of RMS. You must plan to provide access to a specific group of users who will be responsible for managing and approving the addition of CFAS attributes beyond the initial implementation. This group of users will be responsible for developing and enforcing the standards for how attributes are set up for the organization.

Users with access to the base form records will also have access to the relevant CFAS records. You can restrict the access to the CFAS records by using the qualifier function and adding security filtering.

Upgrading CFAS

The CFAS functionality is built in such a way that upgrading to future versions of RMS or applying base patches should require minimal conversion and retrofitting. At a high level, the following are the basic steps for porting CFAS to a newer version:

1. For the basic group sets, groups, and attributes, you must port the data that is used to build the screens over to the updated database. If there are any changes made to the structure of these tables, a small conversion may need to be done to the metadata as part of this effort.
2. Once this has occurred, run the cfagen.ksh batch script again to recreate the database entities in the updated database.
3. Then, port the data from the metadata tables to the updated database tables.
4. If any custom functions were created, you may need to retrofit based on the changes in the latest release. You will also need retrofit any custom changes that were made in the RMS functionality to use the flex attributes into the new code.

A

CFAS Use Cases

Because CFAS is so custom, there is no functionality in RMS—online or batch, that actually uses these attributes to drive any processing. This will require additional customization to RMS. However, leveraging the CFAS functionality to support customizations should make the customization process less complex. This chapter describes some examples of common RMS customizations and how you can leverage the CFAS framework instead, with or without additional custom processing added, to facilitate the business process. This chapter includes the following sections:

External Application Attributes

Because RMS is often the foundation data master for a retailer, it is also often the place where attributes that are needed by other applications are created and maintained, even those that are just informational in RMS. For example, common customizations are to add item, supplier, store, and warehouse attributes to RMS that are then sent to other applications to drive processing in those applications. You can set up and maintain such attributes with CFAS. The integration that is built for this purpose would then be built to use the Views defined at the attribute group or attribute group set to query the information from RMS.

Alternate Hierarchies

Alternate hierarchy functionality is something that is available in the RPAS applications and has often been requested in RMS. You can use the CFAS framework to support this functionality by creating an attribute group titled Alternate Hierarchy and then including attributes such as Division, Group, Department, Class, and Subclass. To set the valid values for each of these attributes, you can choose to use the base merchandise hierarchy tables or define new ones using the Codes table based on your business need.

In case you want more than one alternate hierarchy, you can create an attribute group set called Alternate Hierarchy and then create multiple identical attribute groups for each alternate hierarchy required. By setting up alternate hierarchies in this manner, you can then build reporting based on these hierarchies.

Note: Creating reports or any other functionality that is driven off these alternate hierarchies will require further customization.

Ranging and Grading

The ability to better control grading stores and then using that to range items to the stores based on certain attributes in RMS is also a frequently requested functionality. The CFAS framework provides a way to achieve this ability. You can add a Grade attribute to both stores and items. Then, you can build custom processes to use these attributes to range items to locations, set up replenishment parameters, and so on. Although this will involve further customization, leveraging the CFAS framework will reduce the effort involved.

Electronic Data Interchange (EDI)

For some retailers, there is a requirement to send additional information with the purchase order (PO) using EDI, than the current feature available in RMS. In order to meet this requirement, you can create the custom attributes on the PO header using the CFAS framework, and then make the custom modifications to send these attributes through EDI.

Difference Between CFAS and User Defined Attributes (UDAs)

For items, you now have two different ways (CFAS and UDAs) to create custom attributes without making code changes. The table below highlights some of the key differences that you must consider when identifying attributes that need to be created in RMS.

Feature	CFAS	UDAs
Has explicit data objects in database	Yes - views and staging tables are created using attribute names in the database for querying and uploading purposes, respectively.	No - UDAs are held on UDA tables and accessed by a UDA code.
Can be interfaced to/from external systems	Yes - using views to query the attributes and using the staging tables to upload.	Partially - UDAs can be interfaced, but it requires hard coded values in queries and uploads to ensure the correct UDA mapped in integration.
Available for use in RMS functions	No.	Yes - UDAs can be used for several item related functions, such as building criteria based item lists, mass updates and addition to tickets.
Able to be added by end user	No - requires DBA involvement to run scripts.	Yes.
Can be defaulted from department, class and subclass levels	Partially - functionality allows for a utility to default values, but would require a custom package created to define.	Yes.
Can defined as required	Partially - functionality allows for a utility to default values, but would require a custom package created to define.	Yes - at department, class and/or subclass level.

Feature	CFAS	UDAs
Allows for attribute validation	Partially - simple functionality for minimum/maximum value is supported; the functionality also allows for a utility to validate the attribute (as well as across attributes), but this would require a custom package to be defined.	Partially - LOV UDAs are validated against pre-defined valid values.
Limit to number of attributes that can be defined	No - only 10 group sets can be defined and only 22 attributes can be included in each attribute group, but there is not a limit to the number of attribute groups that can be created.	Yes - maximum of 99,999 UDAs can be defined.
Sent to RDW	No.	Yes.

B

CFAS Database Scripts

This appendix describes the CFAS-specific database scripts. It includes the following:

CFAS Database Create Script

Once the attributes are created using the CFAS Maintenance screens, they are not available for use immediately from the RMS base forms. Once set up, you must then run the cfagen.ksh script to create the relevant database objects for the attributes and activate them for use.

The purpose to make the attributes available only after running this script is to allow for simulation of the newly added or modified attributes without affecting other users accessing the existing CFAS user interface.

During activation, the CFAS Database Create script create the following supporting tables:

- Extended Entity Extension table
- Access views at the Attribute Group or Group Set level
- Optional staging tables to allow data import from external sources

Once these objects are created, the ACTIVE_IND at each level of the CFAS hierarchy is set to 'Y'. Users will now be able to see these changes to the CFAS UI when accessed from the base RMS forms.

You must run this script for the following scenarios:

- New CFAS enablements.
- New group sets or groups for the extended entity.
- New attributes created within the group.
- Modified attributes that affect the object, that includes (but not limited to) the VIEW name or VIEW_COL_NAME changes, data type changes, and so on.

Note: The CFAS Maintenance screens only support modification of a few attribute properties. Modifications to the database definitions of the attribute (the view column or the view including mappings to the extension table) are not supported and not recommended.

In case you want to make such changes, see Modifying Attributes.

Since this script builds or recreates CFAS-related database objects, it is recommended that this script be run by users who own the database schema.

Syntax

```
cfagen.ksh <connect> <Level E for entity or G for group set> <Entity/Group Set ID>
<Gen synonym Y/N> <Drop existing STG table Y/N>
```

Where,

- `<connect>` – use this argument to specify the relevant user name to connect to the database.
- `<Level E for entity or G for group set>` – the level you want to generate. Set the argument to E for entities and G for attribute group sets.
- `<Entity/Group Set ID>` – the entity or attribute group set ID.
- `<Gen synonym Y/N>` – use this argument to indicate whether you want to generate relevant synonyms.
- `<Drop existing STG table Y/N>` – use this argument to indicate whether you want to drop the existing staging tables.

CFAS Load Scripts

Although the CFAS user interface is designed for manual data input, the CFAS framework also allows for bulk loading attributes to the CFAS extension database tables using a load utility. This utility is handy when upgrading from earlier versions of RMS or adding a new attribute with data already existing the system.

The CFAS load script selects the information from the staging table defined at the group set level (CFA_ATTRIB_GROUP_SET.STAGING_TABLE_NAME) and loads the information to the CFA extension table depending on the metadata definitions. The staging table is structured the same way as the easy access view - the attribute names are spelled out instead of the extension column names. This is for the ease of data mapping.

Syntax

```
cfastgload.ksh <connect> <input staging table> <delete rec in staging table Y/N>
[input process id]
```

Where,

- `<connect>` – use this argument to specify the relevant user name/wallet to connect to the database.
- `<input staging table>` – the relevant staging table name to load.
- `<delete rec in staging table Y/N>` – use this argument to indicate whether you want to delete the records in the staging table.
- `<input process id>` – use this argument to specify the relevant process ID.

CFAS Migration Script

The CFAS migration script enables you to migrate or promote CFAS metadata from one environment to another efficiently. You can use this script to migrate updated CFAS metadata from a test environment to a production environment in your implementation without the risk of any manual errors.

This section describes how you can use the CFAS migration script and highlights some important considerations that you must review before running the script. It includes the following topics:

Migrating CFAS Metadata

To migrate or promote the CFAS metadata into the target environment, do the following:

1. Make the CFAS metadata or attributes changes and compile them in the source environment.
2. Run the CFAS migration script specifying the necessary arguments. For more information on the syntax, see Syntax. When you run the script, an output file with the level name and ID is generated. This file includes the merge statements for the metadata recorded you want to promote.
3. Compile the CFAS custom package specification/definition.
4. Run the migration output file in the target environment, and then commit the changes.
5. Run the CFAS Database Create script (cfagen.ksh).
6. Run the output file from the CFAS Database Create script.
7. Compile the CFAS custom package body.

Syntax

```
cfamigrate.ksh <connect> <level> <ID>
```

Where,

- `<connect>` – Use this argument to specify the relevant user name to connect to the database.
- `<Level>` – Use this argument to indicate the CFAS metadata hierarchy level at which the changes were made. The following are the valid level values:
  - **L** for all levels. This creates a complete copy of the CFAS metadata.
  - **E** for entity level. This creates a complete copy of the metadata from entity to attribute, but no codes and record groups.
  - **S** for group set. This gets a snapshot from the group set to attribute.
  - **G** for group. This gets a snapshot from the group to attribute.
  - **A** for attribute. This gets a snapshot of the attribute only.
  - **C** for codes. This gets a snapshot of the code type and detail.

  - **R** for record group. This gets a snapshot of the record group only.
- <ID> – The CFAS metadata hierarchy ID at which the changes were made. This is optional in some levels. Specify an ID based on the following:
  - When the migration level is L, E, C, or R, no ID is required. All the metadata will be exported.
  - When the migration level is S, G, or A, an ID is required. Enter a specific ID to export only that metadata. If no value is entered, all records will be exported.

Note: When you run the script, an output file with the level name and ID is generated. This file includes the merge statements for the metadata recorded you want to promote.

Considerations

Before you run the CFAS migration script, review the following considerations:

- If the migrating metadata are not at the All level and the attribute is using a new Code Detail or Record Group, then those must be migrated first.
- No delete is migrated.
- If the migrated metadata is already available in the target/destination environment, the record will be updated as if it is has been activated regardless of its current status.
- Do not promote any backend update in the source environment with this tool.
- Since the sequence ID for the metadata can be out of sync between environments, the migration tool relies on other means to ensure record uniqueness:
  - For Code Detail, code_type.
  - For Rec Group: rec_group_name.
  - For Entity: base_rms_table.
  - For Group Set: base_rms_table and group_set_view_name.
  - For Group: group_set_view_name and group_view_name.
  - For Attribute: group_view_name and view_col_name.
- If the migrated metadata does not match any of the existing records in the target/destination environment by the above column, then it is considered as new and will be inserted into the target/destination environment.

  For example, consider that an inactive metadata in the source environment was migrated to another environment. Once migrated, some of the columns were changed in the source environment. When migrated again, these columns will be considered as new record in the target/destination environment.

C

CFAS User Interface Validation Routines

This chapter describes the validation routines in the CFAS user interface. It also highlights the simple and complex validations concepts. It includes the following topics:

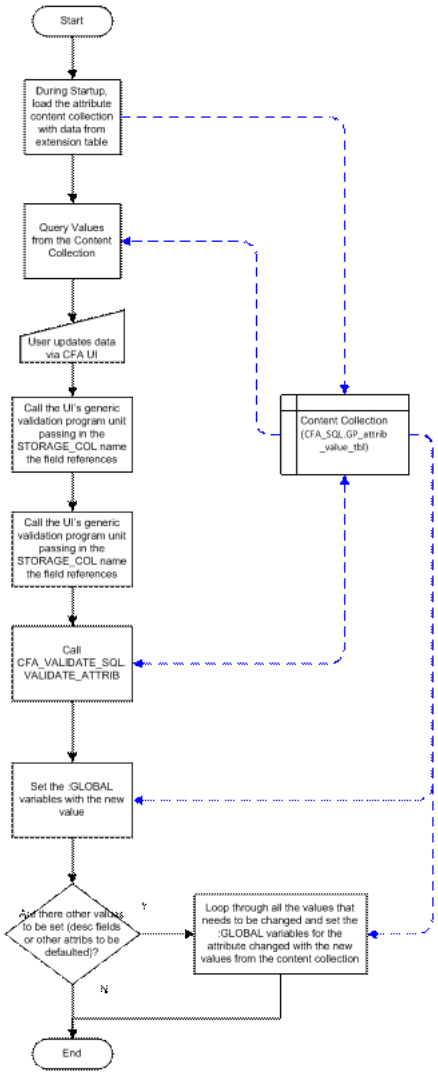
About Validation Routines

For validations requiring more than the basic validations the CFAS framework provides, the CFAS can invoke stored procedures to validate the passed in data in any of the following levels:

- Entity - Entity level validation is the highest level and triggered by the pre-enabled base RMS forms usually via the OK button in the form (to check if there are missing required attributes).
- Group Set - At this level, the validation is triggered when the users exits out of the CFAS user interface to check the data integrity between attributes across all groups within the group set. This validation can optionally be set up in the CFA_ATTRIB_GROUP_SET for each group set.
- Group - This level of validation is triggered when shifting from one attribute group to another (navigating the groups in the attribute group multi-record block). This is optionally set in the CFA_ATTRIB_GROUP metadata table for each group.
- Attribute - This is the lowest level of validation in the CFAS user interface. The validation happens when the user moves out of an attribute field. The validation is optionally defined in the CFA_ATTRIB table.

Starting off with the lowest level validation, the attribute validation works as follows:

Figure C–1 CFAS UI Validation Routine



During startup, the CFAS user interface populates the attribute content (values) collection with the values fetched from the extension table. The user interface uses this collection to populate the physical fields in the user interface via the query procedure of the attribute block.

Once the user updates the field (in case of a field level validation), the WHEN-VALIDATE-ITEM trigger runs and calls the generic FORM_VALIDATION.VALIDATE_FIELD passing in the STORAGE_COL name the field physically maps to in the extension table. This is used as a guide as to determine the metadata information to be used from the attribute configuration collection. At this point the new data is not yet stored to the content collection.

Note: The attribute configuration collection is indexed by VIEW_COL_NAME. To get the right index to the configuration collection, another collection is used to map out the STORAGE_COL_NAME to the VIEW_COL_NAME. This is called the mapping collection (FLEX_UI. LP_strg_vw_tbl).

The VALIDATE_FIELD function then calls the CFA_VALIDATE_SQL.VALIDATE_ATTRIB which does both the simple and complex validations. CFA_VALIDATE_SQL.VALIDATE_ATTRIB gets the metadata information from the configuration collection for the passed attribute (VIEW_COL_NAME) namely, the required and min max information, and if there is complex validation routine.

Simple Validations

For the simple validations, the input value (coming directly from the user interface) is validated against the set values from the metadata. Simple validations checks for the following:

- Data length for VARCHAR2 and NUMBER data types.
- Lowest/Highest allowable value for NUMBER and DATE data types.

Once the input data passes the simple validations, it is then saved to the content collection (but not to the database yet).

Complex Validations

If there is a complex validation defined for the attribute in the metadata, it executes the function dynamically (EXEC_FUNC). The validation functions have only O_error_message as the parameter. The input and output values are defined differently from usual RMS user interface. These rely heavily on the values stored in the content collection.

The following CFAS functions are used to get and set values from this collection:

- CFA_SQL.GET_VALUE - this function is used to retrieve the value of a particular attribute from the content collection. The input to this collection is the VIEW_COL_NAME and returns the value, description if any and the data type. This is used as replacement to the usual input function parameters and preferred rather than using limited generic parameters.
- CFA_SQL.SET_OUTPUT - this function is used to set the value of a particular attribute in the content collection. It can also be used to set the corresponding description fields used by RG type fields. This not only updates the content collection but also builds the return collection. The return collection signals the CFAS UI to update the physical fields on the UI with the values from the content collection. This return collection contains the attributes that were changed by validation function. Only attributes in any group within the attribute group set can be set using this function.

Note: When you choose to set up a validation function at the attribute group level, you must use the following Return function in the validation function. It sets the error field where the validation has failed:

```
CFA_SQL.SET_RETURN_FIELD(
   O_error_message      IN OUT   RTK_ERRORS.RTK_TEXT%TYPE,
   I_field              IN       GP_FIELD_ITEM%TYPE)
```

Where, the `I_field` takes the view column name.

To use this function, you will add lines of code similar to the following:

```
else
      if not CFA_SQL.SET_RETURN_FIELD(O_error_message,'I_field')
then
       return FALSE;
    end if;
```

Basic Sections of the Custom Complex Validation

The body of a validation function can be subdivided into the following three areas:

- **Input Section** - This section acts as the input parameter for the function. The section may have multiple calls to CFA_SQL.GET_VALUE to get the parameters needed for the validation. Values using this function are limited to the header field values (or the CFA_EXT_ENTITY_KEY.KEY_COL) and the attribute fields values in any group within the group set (or the CFA_ATTRIB.VIEW_COL_NAME). If the function requires other values (other than these such as system options or values from the main RMS base table, and so on), the function needs to define a cursor to fetch these information.
- **Validation Logic Section** - This section is where the actual complex validation takes place. The validation logic can be written out directly or make calls to regular RMS package functions.
- **Output Section** - If the validation logic function returns a value that needs to be used to update a field on the form, such as description or other fields that are defaulted, calls to the CFA_SQL.SET_OUTPUT are done in this section.

The following sample code illustrates how a validation function is written:

```
FUNCTION GET_ITEM_DESC (
O_error_message IN OUT RTK_ERRORS.RTK_TEXT%TYPE)
RETURN BOOLEAN AS

   L_program    VARCHAR2(62) := 'CUSTOM_CFA_DEMO_SQL.GET_ITEM_DESC';
   L_field      CFA_SQL.GP_FIELD_ITEM%TYPE := 'ITEM';
   L_item       ITEM_MASTER.ITEM%TYPE;
   L_desc       ITEM_MASTER.ITEM_DESC%TYPE;
   L_data_type  CFA_SQL.GP_DATA_TYPE%TYPE;

BEGIN



   ---------------------------------------------------------------------------
   -- Get keys and/or attribute value inputs in this section.
   ---------------------------------------------------------------------------
   if NOT CFA_SQL.GET_VALUE(O_error_message,
```

```
                          L_item,
                          L_desc, -- initially NULL
                          L_data_type,
                          L_field) then
      return FALSE;
   end if;

   ---------------------------------------------------------------------------
   -- Validation logic in this section. It can be a call to an RMS package
   -- function.
   ---------------------------------------------------------------------------
   if L_desc is NULL then
      if NOT ITEM_ATTRIB_SQL.GET_DESC(O_error_message,
                                      L_desc,
                                      L_item) then
         return FALSE;
      end if;
   end if;

   ---------------------------------------------------------------------------
   -- If there are outputs set each in this section
   ---------------------------------------------------------------------------
   if NOT CFA_SQL.SET_OUTPUT(O_error_message,
                             CFA_SQL.CFA_KEY,
                             L_field,
                             L_item,
                             L_desc) then
      return FALSE;

   end if;
   return TRUE;
EXCEPTION
   when OTHERS then
      O_error_message := SQL_LIB.CREATE_MSG('PACKAGE_ERROR',
                                            SQLERRM,
                                            L_program,
                                            to_char(SQLCODE));

      return FALSE;
END GET_ITEM_DESC;
…
```

The sample function above gets the description for the item. It uses the CFA_SQL.GET_VALUE to get the item value from the content collection. If more parameters are required, several calls to this function are required for each parameter.

The validation logic section can be any business validation written directly in the section or called from another package.

If the validation needs to output something like a description or update another attribute field in any group within the group set, a call to CFA_SQL.SET_OUTPUT can be made. In the above example, the item description is returned by the RMS function ITEM_ATTRIB_SQL.GET_DESC. The CFA_SQL.SET_OUTPUT takes the description value and updates the content collection.

The following example illustrates how other attributes fields are set:

```
FUNCTION CHK_STOCK_SUPPLIER (O_error_message  IN OUT RTK_ERRORS.RTK_TEXT%TYPE)
RETURN BOOLEAN AS

   L_program    VARCHAR2(62) := 'CUSTOM_CFA_DEMO_SQL.CHK_STOCK_SUPPLIER';
```

```
   L_supplier   SUPS.SUPPLIER%TYPE;

   L_field           CFA_SQL.GP_FIELD_ITEM%TYPE := 'SELL_ON';
   L_field_value     CFA_SQL.GP_FIELD_VALUE%TYPE;
   L_sell_qty        NUMBER;
   L_data_type       CFA_SQL.GP_DATA_TYPE%TYPE;
   L_desc            CFA_SQL.GP_FIELD_DESC_VALUE%TYPE;

BEGIN

   ---------------------------------------------------------------------------
   -- Get the current attribute value
   ---------------------------------------------------------------------------
   if NOT CFA_SQL.GET_VALUE(O_error_message,
                            L_field_value,
                            L_desc, -- initially NULL
                            L_data_type,
                            'STOCK_SUPPLIER') then
      return FALSE;
   end if;

   if L_field_value is NOT NULL then
      L_supplier := to_number(L_field_value);
   end if;
   ---
   if NOT CFA_SQL.GET_VALUE(O_error_message,
                            L_field_value,
                            L_desc, -- initially NULL
                            L_data_type,
                            'SELL_QTY') then
      return FALSE;
   end if;

   if L_field_value is NOT NULL then
      L_sell_qty := to_number(L_field_value);
   end if;


   ---------------------------------------------------------------------------
   -- do processing
   ---------------------------------------------------------------------------
   if L_supplier is NOT NULL then
      if NOT SUPP_ATTRIB_SQL.GET_SUPP_DESC(O_error_message,
                                           L_supplier,
                                           L_desc) then
         return FALSE;
      end if;
   end if;



   ---------------------------------------------------------------------------
   -- set output values
   ---------------------------------------------------------------------------
   if NOT CFA_SQL.SET_OUTPUT(O_error_message,
                             CFA_SQL.CFA_EXT,
                             'STOCK_SUPPLIER',
                             L_supplier,
                             L_desc) then
```

```
      return FALSE;
   end if;
   ---
   if L_sell_qty is NULL then
      if NOT CFA_SQL.SET_OUTPUT(O_error_message,
                                CFA_SQL.CFA_EXT,
                                'SELL_QTY',
                                15) then
         return FALSE;
      end if;
   end if;


   return TRUE;
EXCEPTION
   when OTHERS then
      O_error_message := SQL_LIB.CREATE_MSG('PACKAGE_ERROR',
                                            SQLERRM,
                                            L_program,
                                            to_char(SQLCODE));

      return FALSE;
END CHK_STOCK_SUPPLIER;
```

The above example does a couple of things. It sets the description of the stock supplier and sets the default value of the SELL_QTY attribute to 15, if not populated.

Validation of the data does not end here. It is up to the CFAS user interface to handle whatever is returned back by these validation functions.

For attribute validation, a return collection is set if there are fields updated (or defaulted) other than the validated field. This collection is returned to the user interface. The user interface loops through this, sets the individual fields on the form with the new value from the content collection, and updates the*: GLOBAL* variables related to the set fields.

Higher level validations follow the same pattern as the attribute validation with some key differences as described below:

- **Group level**:
  - The validation is triggered when the user changes the attribute group in the multi-record block.
  - Simple validation only checks for required attributes.
  - In some RMS user interfaces, when an error is encountered, the focus is set to the field that failed. CFAS user interface adopts the same functionality. The validation package function returns the error attribute's STORAGE_COL_NAME which maps out to a field on the form. A navigational procedure intended for CFAS (FLEX_UI_WIDGET.GO_ATTRIB_ITEM) is used to set the focus to the user interface field. The usual error message is displayed. To set the return field, use the stored function CFA_SQL.SET_RETURN_FIELD.
- **Group set level**:
  - The validation is triggered when clicking the OK button on the CFAS user interface.
  - Like the group validation, simple validation checks for all required attributes, but at the group set level (all groups within the group set).

  - Unlike the group validation, the group set validation does not set the focus to the error field. The group validation handles this for the current group.

- **Entity level**:
  - Validation is triggered when clicking the OK button on the base RMS form (not the CFAS user interface).
  - Since the entity level validation is run outside the CFAS user interface, no data is loaded to the CFA collection (where all the validation and data manipulation occurs). Consider the following before running an entity level validation:
    - Any qualifier rules must be run before running the validation rules.
    - Since the entity level validation occurs outside the CFAS user interface, the CFA_SQL.GET_VALUE or CFA_SQL.SET_OUTPUT functions cannot be run. It is recommended to use the access views to retrieve the CFAS records.

  These higher level validations also support complex business validations. These stored procedures are written with the same patterns as the attribute level validation.

D

Debugging CFAS

The CFAS framework also includes a built-in tool designed to help you with the debugging of the code. DBG.MSG (used by the CFAS user interface and libraries) and DBG_SQL.MSG (used by the database stored procedures) are procedures used to display and capture debug messages to a log table.

This tool provides the following features:

- Configurable via the database table (DEBUG_CFG).
- Debug messages are only displayed/logged to a table for a specific user defined condition.
- Target specific program units in the user interface (such as P_FORM_STARTUP, WHEN-VALIDATE-ITEM, and so on) or database stored procedures.
- When working with multiple stored procedures or program units, you can switch the object for which you want to see or log messages.

To set the debugging mode for CFAS:

1. In the relevant procedure, enter the SCHEMA name or user name, OBJECT to debug, and optionally set whether the messages are displayed online (when debugging the UI/library) or saved to a log table (DEBUG_LOG). Set at least one option to Y.
2. If the program unit does not have debug messages, strategically insert the debug messages using the DBG.MSG for the user interface and DBG_SQL.MSG for the stored procedures. For DBG.MSG, use as the usual imessage(). Pass in the name of the program unit to debug and the debug message.
3. Compile the code and run the test.

E

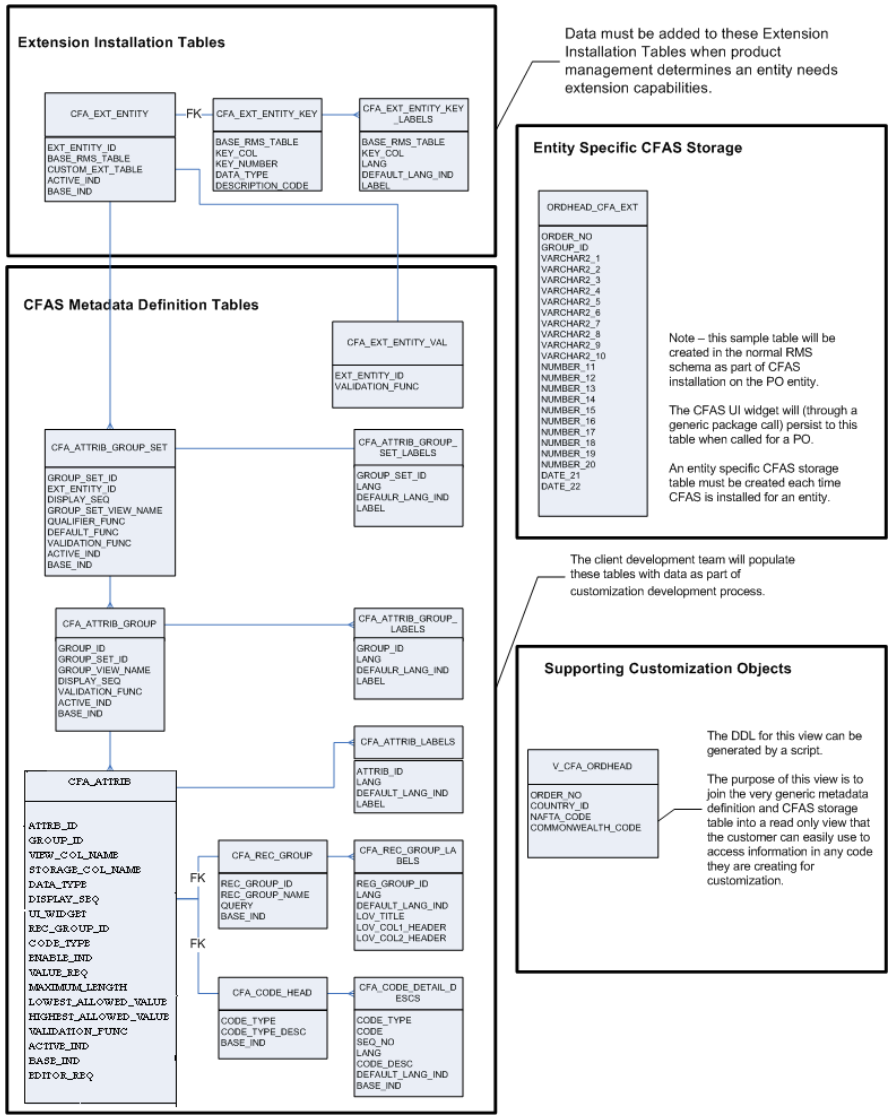
CFAS Table Definitions

The CFAS framework is driven by information stored in the following CFAS-specific database objects. This appendix provides information on the CFAS-specific database objects. It also includes sample data along with each table definition to help you better understand the table definitions.

It includes the following:

- Extension Install Objects
- Custom Metadata Tables
- Entity Specific CFAS Storage
- Supporting Custom Objects

Figure E–1 CFAS-specific Database Objects



Extension Install Objects

These database tables contain information on the entities that are enabled for CFAS. It include the following tables:

- CFA_EXT_ENTITY
- CFA_EXT_ENTITY_KEY
- CFA_EXT_ENTITY_KEY_LABELS

CFA_EXT_ENTITY

This table holds the business object entities that have been extended in the base RMS code. It provides the mapping between extendable business objects (base RMS tables) and custom extension tables. By default, this table contains data that is populated by a seed data script run during the base RMS installation for each entity pre-enabled for customization. You can still populate this table if you want to customize entities not included during the installation. However, changes to the base code using the table are not supported by Oracle.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
EXT_ENTITY_ID	NUMBER(10)	Y	Y	This column holds a generated ID that distinguishes the model extension point.	Create a sequence to generate.
BASE_RMS_TABLE	VARCHAR2(30)	N	Y	This column holds the base RMS table (for example, ordhead, tsfhead, and so on) which the extension refers.	Unique key to ensure no duplicates.
CUSTOM_EXT_TABLE	VARCHAR2(30)	N	Y	This column holds the name of custom extension table related to this model extension. The custom extension table must be created when CFAS is installed for an entity. It is recommended (but not required) that the creator of the metadata append '_CFA_EXT' to the BASE_RMS_TABLE name. This is a recommendation only because some BASE_RMS_TABLE names, when appended with the recommended suffix, may be too long. In those cases, it is recommended that some abbreviation be created for the BASE_RMS_TABLE, and the '_CFA_EXT' suffix is still used.	Unique key to ensure no duplicates.
ACTIVE_IND	VARCHAR2(1)	N	Y	Indicates if the entity is activated and ready for viewing.	Only set via batch. Check constraint (Y,N).

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
BASE_IND	VARCHAR2(1)	N	Y	This indicates that the entity is pre-enabled for customization and supporting code in the base UI are already setup as part of base install.	Check constraint (Y, N).
VALIDATION_FUNCT	VARCHAR2(61)	N	N	This column holds the name (package and function) of code that should be called to validate this entity.	–

Sample Data - CFAS_EXT_ENTITY

The following table illustrates the CFAS_EXT_ENTITY table with sample data.

EXT_ENTITY_ID	BASE_RMS_TABLE	CUSTOM_EXT_TABLE	VALIDATION_FUNCT
3000000001	ORDHEAD	ORDHEAD_CFA_EXT	–
3000000002	ITEM_SUPP_COUNTRY	ITEM_SUP_CTRY _CFA_EXT	–
3000000003	STORE	STORE_CFA_EXT	–
3000000004	SUPS	SUPS_CFA_EXT	–

CFA_EXT_ENTITY_KEY

This table holds the key information for the extended base RMS tables. This information can be derived from system tables, but is instead stored in an application specific table for performance and clarity. This information is used to ensure that the generic CFAS persistence code can map correctly to the entity specific code and table. Data in this table will also be used to determine the number of key fields displayed in the CFAS user interface. By default, this table contains data that is populated by a seed data script run during the base RMS installation for each entity pre-enabled for customization. The values will be automatically populated when using the CFAS Extended Entity Maintenance screen.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
BASE_RMS_TABLE	VARCHAR2(30)	Y	Y	This column holds the base RMS table (for example, ordhead, tsfhead, etc) which the extension refers.	FK to CFA_EXT_ENTITY. BASE_RMS_TABLE.
KEY_COL	VARCHAR2(30)	Y	Y	This column holds the name of the primary key column on the extended BASE_RMS_TABLE.	–

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
KEY_NUMBER	NUMBER	N	Y	This column holds the sequence of the key column. For tables that have a single key, this value will always be 1 and this table will only have one record for the BASE_ RMS TABLE. For tables that have composite primary keys, this column will distinguish which element of the PK is represented and this table will have as many records as there are elements in the composite key.	–
DATA_TYPE	VARCHAR2(10)	N	Y	This column holds the data type of the primary key element.	–
DESCRIPTION_CODE	VARCHAR2(61)	N	N	This column contains the package.function name used by the CFAS UI to get the key value's description	–

Sample Data - CFA_EXT_ENTITY_KEY

The following table illustrates the CFA_EXT_ENTITY_KEY table with sample data.

BASE_RMS_TABLE	KEY_COL	KEY_NUMBER	DATA_TYPE	DESCRIPTION_CODE
ORDHEAD	ORDER_NO	1	NUMBER	
ITEM_SUPP_COUNTRY	ITEM	1	NUMBER	CUSTOM_SQL.GET_ITEM_DESC
ITEM_SUPP_COUNTRY	SUPPLIER	2	NUMBER	CUSTOM_SQL.GET_SUP_DESC
ITEM_SUPP_COUNTRY	ORIGIN_COUNTRY_ID	3	VARCHAR	CUSTOM_SQL.GET_CTRY_DESC
STORE	STORE	1	NUMBER	CUSTOM_SQL.GET_STORE_DESC
SUPS	SUPPLIER	1	NUMBER	CUSTOM_SQL.GET_SUP_DESC

CFA_EXT_ENTITY_KEY_LABELS

This table holds the description that should be used as the label for the key fields in the CFAS user interface header. Records must exist in this table to ensure that fields in the user interface are always labeled. By default, this table contains data that is populated by a seed data script run during the base RMS installation for each entity pre-enabled for customization (for all RMS supported languages).

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
BASE_RMS_TABLE	VARCHAR2(30)	Y	Y	This column holds the base RMS table (for example, ordhead, tsfhead, etc) which the extension refers.	FK to CFA_EXT_ENTITY_ KEYS. BASE_RMS_TABLE.
KEY_COL	VARCHAR2(30)	Y	Y	This column holds the name of the primary key column on the extended BASE_RMS_TABLE.	CFA_EXT_ENTITY_KEYS. KEY_COL.
LANG	NUMBER(6)	Y	Y	This column holds the language that the KEY_DESC is in. It will be used to ensure the appropriate language is shown to any end users accessing the CFAS user interface.	FK to LANG.LANG. UK on BASE_RMS_TABLE, KEY_COL, LANG.
DEFAULT_LANG_IND	VARCHAR2(1)	N	Y	This column indicates that the current records description should be considered the default and displayed in the user interface, if a translation for the end user's language does not exist. This column should be 'Y' for only one language for the BASE_RMS_TABLE and KEY_COL combination.	Check constraint (Y, N).
LABEL	VARCHAR2(255)	N	Y	This value will be displayed on the header of the CFAS user interface to label the value passed from the calling form to the CFAS user interface as the KEY_COL.	–

Sample Data - CFA_EXT_ENTITY_KEY_LABELS

The following table illustrates the CFA_EXT_ENTITY_KEY_LABELS table with sample data.

BASE_RMS_TABLE	KEY_COL	LANG	DEFAULT_LANG_IND	KEY_DESC
ORDHEAD	ORDER_NO	1 (English)	Y	Order
ORDHEAD	ORDER_NO	4 (Spanish)	N	Orden de Compra
ORDHEAD	ORDER_NO	12 (Brazilian Portuguese)	N	Ordem de Compra
ITEM_SUPP_COUNTRY	ITEM	1 (English)	Y	Item
ITEM_SUPP_COUNTRY	ITEM	4 (Spanish)	N	Item (in Spanish)
ITEM_SUPP_COUNTRY	SUPPLIER	1 (English)	Y	Supplier
ITEM_SUPP_COUNTRY	SUPPLIER	4 (Spanish	N	Supplier (in Spanish)
ITEM_SUPP_COUNTRY	ORIGIN_COUNTRY_ID	1 (English)	Y	Origin Country ID
ITEM_SUPP_COUNTRY	ORIGIN_COUNTRY_ID	4 (Spanish)	N	Origin Country ID (in Spanish)

Custom Metadata Tables

The custom metadata database tables contain all the information required to display and capture actual data on the extended entities. They include:

- CFA_ATTRIB_GROUP_SET
- CFA_ATTRIB_GROUP_SET_LABELS
- CFA_ATTRIB_GROUP
- CFA_ATTRIB_GROUP_LABELS
- CFA_ATTRIB
- CFA_ATTRIB_LABELS
- CFA_REC_GROUP
- CFA_REC_GROUP_LABELS
- CFA_CODE_HEAD
- CFA_CODE_DETAIL_DESCS

CFA_ATTRIB_GROUP_SET

This table holds the metadata that defines attribute group sets for all entities. Each extended entity can have at most 10 group sets which are accessible from the Options menu of the relevant user interface. Attribute group sets represent a higher grouping of attributes that are functionally or feature-wise (in case of add-on packs) related.

Access to the group set may be controlled via rules. These rules are executed via function called when the option menu is activated. Only a function can be defined per group but the function can have any rule. See examples setting up rules (TBD).

A validation function can be defined to validate combinations of attributes within the group set spanning across multiple attribute groups.

Column	Datatype	Primary Key?	Required?	Description	Notes
GROUP_SET_ID	NUMBER(10)	Y	Y	This column holds id of the set where the group belongs to.	System Generated.
EXT_ENTITY_ID	NUMBER(10)	N	Y	This column holds the ID of the entity where the set belongs to.	FK to CFA_EXT_ ENTITY.EXT_ENTITY_ID.
GROUP_SET_VIEW_ NAME	VARCHAR2(30)	N	Y	This column holds the name of the database view that will be generated to make access to user entered data easier.	–
DISPLAY_SEQ	NUMBER	N	Y	This column holds the order the attribute group set displayed in on the CFAS user interface when multiple group sets exist for the entity.	Check Constraint (1-10). Unique across an entity.
QUALIFIER_FUNC	VARCHAR2(61)	N	N	This column holds the name (package and function) of code that should be called to check if required information is supplied from the base user interface to access the attributes within the group set (determines if the group set is enabled on the user interface). The inputs and outputs of this function code are tightly controlled.	–

Column	Datatype	Primary Key?	Required?	Description	Notes
DEFAULT_FUNC	VARCHAR2(61)	N	N	This column holds the name (package and function) of code that should be called on startup of the CFAS user interface to pre-populate attribute fields with default values (can be in any group within the set). The inputs and outputs of this attribute group set level default code are tightly controlled.	–
VALIDATION_FUNC	VARCHAR2(61)	N	N	This column holds the name (package and function) of code that should be called to validate this attribute group set. The inputs and outputs of this attribute group set level validation code are tightly controlled.	–
STAGING_TABLE_ NAME	VARCHAR2(30)	N	N	This column holds the name of the staging area where data from an external source can be stored and exported to the CFA extension table linked to this group set.	–
ACTIVE_IND	VARCHAR2(1)	N	Y	This column indicates whether the group set is visible in the base user interface menu. Used for simulation purposes.	Check Constraint (Y, N).
BASE_IND	VARCHAR2(1)	N	Y	This column indicates if the attribute group set is defined by Oracle. Oracle defined group sets cannot be further customized.	Check Constraint (Y, N).

Sample Data - CFA_ATTRIB _GROUP_SET

The following table illustrates the CFA_ATTRIB _GROUP_SET table with sample data.

> **Note:** Grouping the attribute groups into attribute group sets that are logical requires local domain knowledge. The examples below are meant for illustration purposes only.

GROUP_ SET_ID	EXT_ENTITY_ ID	DISP LAY _ SEQ	ACTI VE_ IND	GROUP_SET_VIEW_ NAME	QUALIFIER_FUNC	DEFAULT_ FUNC	VALIDATION _FUNC
1000000001	3000000001	1	Y	V_CFA_PO_CUSTOM		–	–
1000000002	3000000001	2	Y	V_CFA_PO_TELCO	TELCO_SQL.CHK_TELCO_ACCESS	–	–
1000000003	3000000002	1	Y	V_CFA_ISC_TELCO	TELCO_SQL.CHK_TELCO_ACCESS	–	–

CFA_ATTRIB_GROUP_SET_LABELS

This table holds the language specific descriptions that will be used to depict each specific group set in the attribute group set section of the CFAS UI. At least one record must exist on this table for each attribute group set.

Column	Datatype	Primary Key?	Required?	Description	Notes
GROUP_SET_ID	NUMBER(10)	Y	Y	This column holds a generated ID that distinguishes the custom attribute group set.	FK to CFA_ATTRIB _ GROUP_SET. GROUP_SET_ ID.
LANG	NUMBER(6)	Y	Y	This column holds the language the of the user interface description.	FK to LANG.LANG.
DEFAULT_ LANG_IND	VARCHAR2(1)	N	Y	This column indicates that the current records description should be considered the default and displayed in the user interface, if a translation for the end user's language does not exist. This column should be 'Y' for only one language for the GROUP_ID.	Check constraint (Y, N).
LABEL	VARCHAR2(255)	N	Y	This column holds the text that will be used to identify the attribute group on the CFAS UI.	–

Sample Data - CFA_ATTRIB_GROUP_LABELS

The following table illustrates the CFA_ATTRIB_GROUP_LABELS table with sample data.

Note: All text in the table below are for illustration purposes only.

GROUP_SET_ID	LANG	DEFAULT_LANG_IND	LABEL
1000000001	1	Y	Purchase Order Custom Attributes
1000000002	1	Y	Purchase Order Telecom Attributes
1000000003	1	Y	Item Supplier Country Telecom Attributes
1000000001	4	Y	Purchase Order Custom Attributes (In Spanish)
1000000002	4	Y	Purchase Order Telecom Attributes (In Spanish)
1000000003	4	Y	Item Supplier Country Telecom Attributes (In Spanish)

CFA_ATTRIB_GROUP

This table holds the metadata that defines attributes groups for group sets for all entities.

Attribute groups define the grouping of attributes in the user interface widget. Each attribute group is limited to 22 possible attributes. 10 of these attributes can be character data, 10 attributes can be numeric data and two attributes can be date data. If you need to add 12 attributes to an entity, and each attribute is a number, you will need to split the attributes into two attribute groups.

A validation function can be defined to validate combinations of attributes within the group.

Column	Datatype	Primary Key?	Required?	Description	Notes
GROUP_ID	NUMBER(10)	Y	Y	This column holds a generated ID that distinguishes the custom attribute group.	Create a sequence to generate these numbers.
GROUP_SET_ID	NUMBER(10)	N	Y	This column holds id of the set where the group belongs to.	FK to CFA_EXT_ENTITY_GROUP_SET. GROUP_SET_ID.
GROUP_VIEW_NAME	VARCHAR2(30)	N	Y	This column holds the name of the database view that will be generated to make access to user entered data easier.	–

Column	Datatype	Primary Key?	Required?	Description	Notes
DISPLAY_SEQ	NUMBER	N	Y	This column holds the order the attribute groups will be displayed in on the CFAS UI when multiple groups exist for a single attribute group set.	
VALIDATION_ FUNC	VARCHAR2(61)	N	N	This column holds the name (package and function) of code that should be called to validate this attribute group. The inputs and outputs of this attribute group level validation code are tightly controlled.	–
ACTIVE_IND	VARCHAR2(1)	N	Y	This column indicates whether the group is visible in the CFAS user interface when accessed from the base user interface. Used for simulation purposes.	Check Constraint (Y, N).
BASE_IND	VARCHAR(1)	N	Y	This column indicates if the attribute group is defined by Oracle. Oracle defined groups cannot be further customized.	Check Constraint (Y, N).

Sample Data - CFA_ATTRIB _GROUP

The following table illustrates the CFA_ATTRIB _GROUP table with sample data.

Note: Grouping the attributes into attribute groups that are logical requires local domain knowledge. The division of these attributes into attribute groups in the table below is for illustration purposes only; it does not reflect the local domain knowledge that will be required to create logical and usable CFAS metadata.

GROUP_ID	GROUP_SET_ID	DISPLAY_SEQ	ACTIVE_IND	GROUP_VIEW_NAME	VALIDATION_ FUNC
9000000001	1000000001	1	Y	V_CFA_PO_IMPORT	–
9000000002	1000000001	2	Y	V_CFA_PO_SHIPPING	–
9000000004	1000000001	3	N	V_CFA_PO_BILLING	–
9000000006	1000000002	1	Y	V_CFA_PO_TELCO_ATTRIB	–

GROUP_ID	GROUP_SET_ID	DISPLAY_SEQ	ACTIVE_IND	GROUP_VIEW_NAME	VALIDATION_ FUNC
9000000007	1000000002	2	Y	V_CFA_PO_TELCO_CONTACT	–
9000000008	1000000003	1	Y	V_CFA_ISC_ITEM_RESTRICT	–
9000000009	1000000003	2	Y	V_CFA_ISC_INV	–

CFA_ATTRIB_GROUP_LABELS

This table holds the language specific descriptions that will be used to show each specific group in the attribute groups section of the CFAS user interface. At least one record must exist on this table for each attribute group.

Column	Datatype	Primary Key?	Required?	Description	Notes
GROUP_ID	NUMBER(10)	Y	Y	This column holds a generated ID that distinguishes the custom attribute group.	FK to CFA_ATTRIB _ GROUP. GROUP_ID.
LANG	NUMBER(6)	Y	Y	This column holds the language the of the UI description.	FK to LANG.LANG
DEFAULT_ LANG_IND	VARCHAR2(1)	N	Y	This column indicates that the current records description should be considered the default and displayed in the user interface, if a translation for the end user's language does not exist. This column should be 'Y' for only one language for the GROUP_ID.	Check constraint (Y, N).
LABEL	VARCHAR2(255)	N	Y	This column holds the text that will be used to identify the attribute group on the CFAS user interface.	–

Sample Data - CFA_ATTRIB_GROUP_LABELS

The following table illustrates the CFA_ATTRIB_GROUP_LABELS table with sample data.

Note: All text in the table below are for illustration purposes only.

GROUP_ID	LANG	DEFAULT_LANG_IND	LABEL
9000000001	1	Y	Purchase Order Import Attributes
9000000002	1	Y	Purchase Order Shipping Attributes
9000000004	1	Y	Purchase Order Billing Attributes
9000000006	1	Y	Purchase Order Telecom Attributes
9000000007	1	Y	Purchase Order Telecom Contact Information
9000000008	1	Y	Item Supplier Country Restrictions
9000000009	1	Y	Item Supplier Country Telecom Attributes
9000000001	4	N	PO Import Attributes (In Spanish)
9000000002	4	N	PO Shipping Attributes (In Spanish)
9000000004	4	N	PO Billing Attributes (In Spanish)
9000000006	4	N	PO Telecom Attributes (In Spanish)
9000000007	4	N	PO Telecom Contact Information (In Spanish)
9000000008	4	N	Item Restrictions (In Spanish)
9000000009	4	N	Item Telecom Attributes (In Spanish)

CFA_ATTRIB

This table holds the metadata that defines custom attributes for all entities. The information stored in this table tells how each attribute is stored, basic data restrictions, and how the attribute is displayed in the CFAS user interface.

There are check constraints/unique constraints to ensure that each group contains no more than the prescribed 22 attributes, and that no more than 10 of these attributes are numbers, no more than 10 attributes are chars, and no more than 2 attributes are dates.

Column	Datatype	Primary Key?	Required?	Description	Notes
ATTRIB_ID	NUMBER (10)	Y	Y	This column holds the generated ID of the attribute.	Create a sequence to generate these numbers.
GROUP_ID	NUMBER (10)	N	Y	This column holds the ID of the group this attribute belongs to.	FK to CFA_ATTRIB_ GROUP.
VIEW_COL_NAME	VARCHAR2(30)	N	Y	This column holds the description of the attribute. This description is limited to 30 characters so it can be referenced in the easy access views. This text string should not contain spaces or special characters.	Character string should be in all caps. Unique across the entity.
STORAGE_COL_ NAME	VARCHAR2(11)	N	Y	This column holds the text string that corresponds to the storage column on the entity specific CFAS Storage/Extension table.	Check constraint - value must be in the list: VARCHAR2_1, VARCHAR2_2, VARCHAR2_3, VARCHAR2_4, VARCHAR2_5, VARCHAR2_6, VARCHAR2_7, VARCHAR2_8, VARCHAR2_9, VARCHAR2_10, NUMBER_ 11, NUMBER_12, NUMBER_13, NUMBER_14, NUMBER_15, NUMBER_16, NUMBER_17, NUMBER_18, NUMBER_19, NUMBER_20, DATE_21, DATE_22

Column	Datatype	Primary Key?	Required?	Description	Notes
DISPLAY_SEQ	NUMBER(2)	N	Y	This column determines the order in which the attribute will be displayed on the CFAS user interface.	Check Constraint - number must be between 1 and 10 if data type is VARCHAR2, 11 and 20 if data type is NUMBER or 21 and 22 if data type is DATE.
DATA_TYPE	VARCHAR2(8)	N	Y	This column holds the data type of the attribute. Valid values are limited to VARCHAR2, NUMBER and DATE.	Check Constraint - 'VARCHAR2', 'NUMBER', 'DATE'.
UI_WIDGET	VARCHAR2(2)	N	Y	This column holds the user interface item type that should be displayed on the Custom Data Entry user interface. Valid values are TI (text item), RG (record group), LI (list item), CB (check box) and DT (date). Note that if the check box item type is used, the checked value will always be 'Y' and the unchecked value will always be 'N'. Default value is 'N' (unchecked) if the attribute is required.	Check Constraint - valid values are TI, RG, LI, CB, DT.
REC_GROUP_ID	NUMBER (10)	N	N	This column holds the ID of the record group associated with the attribute. Record groups are only needed when the attribute uses the record group ('RG') UI item type. Record groups are defined on the CFA_REC_GROUP table. CFA record groups must be defined before an attribute using the record group is defined.	FK to CFA_REC_GROUP. REC_GROUP_ID.
CODE_TYPE	VARCHAR2(4)	N	N	This column holds the specific custom code type associated with the attribute. Code types are only needed when the attribute uses the list item ('LI') UI item type.	FK to CFA_CODE_HEAD. CODE_TYPE.
ENABLE_IND	VARCHAR2(1)	N	Y	This column determines if the field is disabled (display only) or editable. The value of the attribute should be sourced from CFA_ATTRIB_GROUP_ SET.DEFAULT_FUNC if the attribute is disabled.	Check Constraint - valid values are 'Y', 'N'.

Column	Datatype	Primary Key?	Required?	Description	Notes
VALUE_REQ	VARCHAR2(1)	N	Y	This column determines if the field is required to be not null when a record is created in the custom extension storage table.	Check Constraint - valid values are 'Y', 'N'.
MAXIMUM_ LENGTH	NUMBER(20)	N	N	This column determines the maximum valid length the user can enter for field on the UI. A value is recommended for Char and Number attributes, and not valid for date attributes.	–
LOWEST_ ALLOWED_VALUE	NUMBER (20, 4)	N	N	This value determines if the lowest numeric value that can be entered in the field. This value is optional and should only be populated when the data type of the attribute is NUMBER or DATE.	–
HIGHEST_ ALLOWED_VALUE	NUMBER (20, 4)	N	N	This value determines if the highest numeric value that can be entered in the field. This value is optional and should only be populated when the data type of the attribute is NUMBER or DATE.	–
VALIDATION_ FUNC	VARCHAR2(20)	N	N	This column holds the name (package and function) of code that should be called to validate this attribute.	For an example of the type of validation code that could be written for field level validation, see CFAS User Interface Validation Routines.
ACTIVE_IND	VARCHAR2(1)	N	Y	This column indicates whether the attribute is displayed in the CFAS user interface when accessed from the base user interface. Used for simulation purposes.	Check Constraint - Y,N.
BASE_IND	VARCHAR2(1)		Y	This column indicates if the attribute is Oracle defined. Oracle defined attributes can not be further customized.	Check Constraint - Y,N.
EDITOR_REQ	VARCHAR2(1)		Y	This column indicates whether the editor is required for the attribute.	Check Constraint - Y,N.
Unique constraint on group_id and storage_col_name (to ensure metadata is not defined that attempts to map two attributes to the same slot on the entity extension table)					

Sample Data - CFA_ATTRIB

The following table illustrates the CFA_ATTRIB table with sample data.

Note: From the examples above, Group id 90000000001 extends ordhead for additional PO import attributes. Groups 9000000002 extends store for additional PO shipping attributes.

If more extensive validation is needed for any single attribute, the name of a custom specific PL/SQL function can be specified in VALIDATION_FUNC column on this table. Any validations specified at the attribute level will fire as item level validation (in Oracle forms terms, in when validate item triggers).

ATTRIB_ID	GROUP_ID	DISPLAY_SEQ	ACTIVE_IND	VIEW_COL_NAME	STORAGE_COL_NAME	DATA_TYPE	UI_WIDGET	ENABLE_IND	VALUE_REQ	MAXIMUM_LENGTH	LOWEST_ALLOWED_VALUE	HIGHEST_ALLOWED_VALUE	REC_GROUP_ID	CODE_TYPE	VALIDATION_FUNC
9876001	9000000001	1	Y	ENTRY_METHOD	VARCHAR2_1	VARCHAR2	LI	Y	N	4	–	–	–	ENTR	–
9876002	9000000001	2	Y	PORT	NUMBER_11	NUMBER	TI	Y	N	10	–	–	–	–	–
9876003	9000000002	1	Y	SHIP_TYPE	VARCHAR2_1	VARCHAR2	LI	Y	N	4	–	–	–	–	–
9876004	9000000003	2	Y	SHIP_CAP	NUMBER_11	NUMBER	TI	Y	N	10	0	9999	–	–	–
9876005	9000000004	3	Y	BILL_TYPE_CODE	NUMBER_12	NUMBER	RG	Y	N	10	–	–	77777	–	–
9876006	9000000004	4	Y	BILL_DATE	DATE	DATE	DT	N	N	–	–	–	–	–	–

Additional Note about Validation

If additional validation is required to ensure that two attributes in the same group are somehow compatible, the inter-attribute group validation code can be listed in CFA_ATTRIB_GROUP.VALIDATION_FUNC.

If additional validation is required to ensure that groups of attributes are somehow compatible, the validation code can be listed in CFA_ATTRIB_GROUP_SET.VALIDATION_FUNC. This type of validation could be used in cases where attributes are required in either one group or another (for example, the partner is either a federal taxpayer or a local taxpayer).

CFA_ATTRIB_LABELS

This table holds descriptions that will be used to label the attribute on the CFAS user interface.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
ATTRIB_ID	NUMBER(10)	Y	Y	This column holds the attribute id of the attribute being described.	FK to CFA_ ATTRIB.ATTRIB_ID.
LANG	NUMBER(6)	Y	Y	This column holds the language the description is in.	FK to LANG.LANG.
DEFAULT_ LANG_IND	VARCHAR2(1)	N	Y	This column indicates that the current records description should be considered the default and displayed in the user interface, if a translation for the end user's language does not exist. This column should be 'Y' for only one language for the GROUP_ID.	Check constraint (Y, N).
LABEL	VARCHAR2(255)	N	Y	This column holds the text that will be used to label the attribute on the CFA UI.	–

Sample Data - CFA_ATTRIB_LABELS

The following table illustrates the CFA_ATTRIB_LABELS table with sample data.

Note: All text in the table below are for illustration purposes only.

ATTRIB_ID	LANG	DEFAULT_LANG_IND	DESC
32123323	1 (English)	Y	–

CFA_REC_GROUP

This table contains the source query for record groups used by LOVs in the CFAS UI.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
REC_GROUP_ID	NUMBER (10)	Y	Y	This column holds generated ID of the Custom record group.	Create a sequence to generate these IDS.
REC_GROUP_NAME	VARCHAR2(30)	N	Y	This column holds the name of the record group.	–
QUERY	VARCHAR2(2000)	N	Y	This column holds the query that will be built or executed when the record group is associated with an attribute and displayed on the CFAS UI. The query must adhere to several rules: It must be a complete, valid query that is capable of executing in a SQL*Plus session. It must follow the translation standards. It can have no more than two columns in the SELECT portion of its statement, one ID column and one description column. Bind variables to limit the query can be used. The variable names should be the same as that with the CFA_ATTRIB.VIEW_COL_NAME or CFA_EXT_ENTITY_KEY.KEY_COL Ensure that any custom record groups have appropriate, functioning queries.	–
BASE_IND	VARCHAR(1)	N	Y	This indicates that the record group is pre-enabled for customization and supporting code in the base UI are already setup as part of base install.	Check constraint (Y, N).
QUERY_TYPE	VARCHAR(2)	N	Y	This indicates the query type (simple or complex) set up for the record group.	–
TABLE_NAME	VARCHAR2(30)	N	N	This indicates the relevant table name used to build the simple query for the record group.	–
COLUMN_1	VARCHAR2(30)	N	N	This indicates the column name of the first column to be used in building the simple query for the record group.	–

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
COLUMN_2	VARCHAR2(30)	N	N	This indicates the column name of the second column to be used in building the simple query for the record group.	–
WHERE_COL_1	VARCHAR2(30)	N	N	This indicates the column name of the first column used in the WHERE clause of the simple query for the record group.	–
WHERE_OPERATOR_1	VARCHAR2(6)	N	N	This indicates the operator for the first column used in the WHERE clause of the simple query for the record group.	–
WHERE_COND_1	VARCHAR2(120)	N	N	This indicates the condition set for the first column used in the WHERE clause of the simple query for the record group.	–
WHERE_COL_2	VARCHAR2(30)	N	N	This indicates the column name of the second column used in the WHERE clause of the simple query for the record group.	–
WHERE_OPERATOR_2	VARCHAR2(6)	N	N	This indicates the operator for the second column used in the WHERE clause of the simple query for the record group.	–
WHERE_COND_2	VARCHAR2(120)	N	N	This indicates the condition set for the second column used in the WHERE clause of the simple query for the record group.	–
COL_1_DATA_TYPE	VARCHAR2(106)	N	N	This indicates the data type of the first column.	–
COL_1_DATA_LENGTH	NUMBER(10)	N	N	This indicates the length of data set up for the first column.	–

Sample Data - CFA_REC_GROUP

The following table illustrates the CFA_REC_GROUP table with sample data.

REC_GROUP_ID	REC_GROUP_NAME	QUERY	BASE_IND	QUERY_TYPE	TABLE_NAME	COLUMN_1	COLUMN_2	WHERE_COL_1	WHERE_OPERATOR_1	WHERE_COND_1	WHERE_COL_2	WHERE_OPERATOR_2	WHERE_COND_2	COL_1_DATA_TYPE	COL_1_DATA_LENGTH
77777	CITYDESC	select city_desc, city from city where get_primary_lang = get_user_lang and country_id = USA;	N	SIMPLE	city	city_desc	city	get_primary_lang	=	get_user_lang	country_id	=	USA	-	-

CFA_REC_GROUP_LABELS

This table holds the column labels that should be displayed to end users when a specific record group list of values is invoked from the CFAS user interface.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
REC_GROUP_ID	NUMBER(10)	Y	Y	This column holds generated ID of the custom record group.	FK to CFA_REC_GROUP.REC_GROUP_ID.
LANG	NUMBER(6)	Y	Y	This column holds the language the of the user interface description.	FK to LANG.LANG.
DEFAULT_LANG_IND	VARCHAR2(1)	N	Y	This column indicates that the current records description should be considered the default and displayed in the user interface, if a translation for the end user's language does not exist. This column should be 'Y' for only one language for the REC_GROUP_ID.	Check constraint (Y, N).

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
LOV_TITLE	VARCHAR2(255)	N	Y	This column holds the language specific value for the LOV title. This value will be displayed in the LOV header when the record group is invoked from the CFAS user interface.	–
LOV_COL1_ HEADER	VARCHAR2(255)	N	Y	This column holds the language specific value for the first LOV column header. This value will be displayed in the LOV header when the record group is invoked from the CFAS user interface.	–
LOV_COL2_ HEADER	VARCHAR2(255)	N	Y	This column holds the language specific value for the second LOV column header. This value will be displayed in the LOV header when the record group is invoked from the CFAS user interface.	–

Sample Data - L10N_REC_GROUP_DESCS

The following table illustrates the L10N_REC_GROUP_DESCS table with sample data.

L10N_REC_ GROUP_ID	LANG	DEFAULT_LANG_IND	LOV_TITLE	LOV_COL1_HEADER	LOV_COL2_HEADER
77777	1	Y	List of Cities	City Description	City
77777	4	N	List of Cities (In Spanish)	City Description (In Spanish)	City (In Spanish)

CFA_CODE_HEAD

This table holds the code types that will be used as the basis for defining list items in the CFAS UI.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
CODE_TYPE	VARCHAR2(4)	Y	Y	This column holds the distinct code type. This code type can be related to a list item on the CFAS user interface.	–

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
CODE_TYPE_ DESC	VARCHAR2(120)	N	Y	This column holds a description of the code type. The code type description is never displayed to end users, so no translation is necessary. This data exists only to make it easier for you to ensure that you have selected the correct CODE_TYPE when creating the metadata for a list item CFAS attribute.	–

Sample Data - CFA_CODE_HEAD

The following table illustrates the CFA_CODE_HEAD table with sample data.

CODE_TYPE	CODE_TYPE_DESC
BILT	Bill Types

CFA_CODE_DETAIL_DESCS

This table holds the code/descriptions within a code type that will displayed as individual choices within the list items on the CFA UI.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
CODE_TYPE	VARCHAR2(4)	Y	Y	This column holds the distinct code type. This code type can be related to a list item on the CFAS user interface.	FK to CFA_CODE_HEAD. CODE_TYPE.
CODE	VARCHAR2(6)	Y	Y	This column holds a code within the code type. This code will be an individual item within the list on the CFAS user interface widget.	–
SEQ_NO	NUMBER(4)	N	Y	This column determines the order the code values will be displayed within the list.	Unique within a code_type and lang.
LANG	NUMBER(6)	Y	Y	This column defines the language of the code description.	FK to LANG.LANG.

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
DEFAULT_ LANG_IND	VARCHAR2(1)	N	Y	This column indicates that the current records description should be considered the default and displayed in the user interface, if a translation for the end user's language does not exist. This column should be 'Y' for only one language for the CODE_TYPE/ CODE combination.	Check constraint (Y, N).
CODE_DESC	VARCHAR2(40)	N	Y	This column holds the text value that will be displayed within the list to the end user.	–

Sample Data - CFA_CODE_DETAIL_DESCS

The following table illustrates the CFA_CODE_DETAIL_DESCS table with sample data.

CODE_TYPE	CODE	SEQ_NO	LANG	DEFAULT_LANG_IND	CODE_DESC
BILT	ABC	1	1	Y	ABC Description
BILT	BCD	2	1	Y	BCD Description
BILT	CDE	3	1	Y	CDE Description
BILT	DEF	4	1	Y	DEF Description
BILT	ABC	1	4	N	ABC Description (In Spanish)
BILT	BCD	2	4	N	BCD Description (In Spanish)
BILT	CDE	3	4	N	CDE Description (In Spanish)
BILT	DEF	4	4	N	DEF Description (In Spanish)

Entity Specific CFAS Storage

Each time a business entity is extended for customization, an appropriate entity specific CFAS storage table must be created. Creating this table is part of the installation process. This table will be part of the base RMS data model for those entities that are pre-enabled. Each time you extend a business entity (other than the pre-enabled ones) for customization, you must create a relevant entity specific CFAS storage table.

The general structure of this table is as follows:

- Primary Key of BASE_RMS_TABLE
- Group_id

- 10 columns that can hold attributes of the VARCHAR2 data type, named VARCHAR2_1, VARCHAR2_2, VARCHAR2_3, through to VARCHAR2_10. These columns must each be VARCHAR2(250).
- 10 columns that can hold attributes of the NUMBER data type, named NUMBER_11, NUMBER_12, NUMBER_13, through to NUMBER_20.
- Two columns that can hold attributes of type DATE, named DATE_21 and DATE_22.

These column names do not have business meaning to the end user, but are generic enough to store data for most business requirements. Note that most custom specific code will not be written directly against these entity-specific extension tables, but instead around access views. The access views join to the metadata definition tables to provide a structure that has far more business meaning.

As an example, if the goal is to add extensions to purchase orders, there should be a record on CFA_EXT_ENTITY.

EXT_ENTITY_ID	BASE_RMS_TABLE	CUSTOM_EXT_TABLE
3000000001	ORDHEAD	ORDHEAD_CFA_EXT

The structure of ORDHEAD_CFA_EXT should then be:

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
ORDER_NO	NUMBER(8)	Y	Y	This column holds the PO this extended data is associated with.	FK to ordhead.order_no.
GROUP_NO	NUMBER(10)	Y	Y	This column holds the attribute group id that this extended data is associated with. The logical business meaning of the VARCHAR_ , NUMBER_ and DATE_ columns on this table are determined by the metadata defined for this attribute.	FK to CFA_ATTRIB_ GROUP.GROUP_NO.
VARCHAR2_1	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_1 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_2	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_2 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
VARCHAR2_3	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_3 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_4	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_4 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_5	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_5 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_6	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_6 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_7	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_7 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_8	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_8 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_9	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_9 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
VARCHAR2_10	VARCHAR2(250)	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references VARCHAR2_10 as it's CFA_ATTRIB.STORAGE_COL_ NAME.	–
NUMBER_11	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_1 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
NUMBER_12	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_2 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_13	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_3 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_14	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_4 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_15	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_5 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_16	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_6 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_17	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_7 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_18	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_8 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_19	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_9 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
NUMBER_20	NUMBER	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references NUMBER_10 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–

Column	Datatype	Primary Key?	Required?	Description	Keys/Notes
DATE_21	DATE	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references DATE_1 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–
DATE_22	DATE	N	N	This column holds data related to the attribute defined on the CFA_ATTRIB table that references DATE_2 as it's CFA_ATTRIB.STORAGE_COL_NAME.	–

If more than 10 varchar2 attributes, 10 number attributes, or two date attributes are required, you must define additional attribute groups.

Attribute groups also determine how the information is grouped on the CFAS user interface. When designing the CFAS metadata, it is also important to consider the advantages of grouping attributes into multiple functionally related groups from an entity or group set combination.

Supporting Custom Objects

The CFAS framework uses the following supporting custom objects that are generated based on the definitions in the Extension Installation and CFAS Metadata tables. When you modify the metadata, the access views and staging tables will need to be generated again:

- CFAS Access Views
- CFAS Staging Table

These optional objects are business representation of the data that is to be stored in the generic CFAS extension tables.

CFAS Access Views

The main purpose of CFAS access views is to simplify the data access to the metadata driven extension tables. It is recommended that custom code or external systems query the CFAS data from these views rather than directly accessing the CFAS extension tables (depending on performance requirements).

The view names are stored at two levels, at the group set level and at the group level. At the group set level, the names are stored in CFA_ATTRIB_GROUP_SET.GROUP_SET_VIEW_NAME and at the group level in CFA_ATTRIB_GROUP.GROUP_VIEW_NAME. Both these views will have column names defined in CFA_ATTRIB.VIEW_COL_NAME.

From the metadata definition examples earlier, the following tables illustrate the view and column name storage in the relevant tables.

CFA_ATTRIB_GROUP_SET

GROUP_SET_ID	EXT_ENTITY_ID	DISPLAY_ORDER	GROUP_SET_VIEW_NAME
1000000001	3000000001 (ORDHEAD)	1	V_CFA_PO_CUSTOM
1000000003	3000000002 (ITEM_SUPP_COUNTRY)	1	V_CFA_ISC_TELCO

CFA_ATTRIB_GROUP

GROUP_ID	GROUP_SET_ID	GROUP_VIEW_NAME
9000000001	1000000001	V_CFA_PO_IMPORT
9000000002	1000000001	V_CFA_PO_SHIPPING
9000000004	1000000001	V_CFA_PO_BILLING
9000000008	1000000003	V_CFA_ISC_ITEM_RESTRICT
9000000009	1000000003	V_CFA_ISC_INV

CFA_ATTRIB

The following table displays the subset of columns used in the view definition.

ATTRIB_ID	GROUP_ID	ACTIVE_IND	VIEW_COL_NAME	STORAGE_COL_NAME	DATA_TYPE
9876001	9000000001	Y	ENTRY_METHOD	VARCHAR2_1	VARCHAR2
9876002	9000000001	Y	PORT	NUMBER_11	NUMBER
9876003	9000000002	Y	SHIP_TYPE	VARCHAR2_1	VARCHAR2
9876004	9000000003	Y	SHIP_CAP	NUMBER_11	NUMBER
9876005	9000000004	Y	BILL_CODE	NUMBER_12	NUMBER
9876006	9000000004	Y	BILL_DATE	DATE	DATE

The structure of the sample view at the group view set will be:

```
V_CFA_PO_CUSTOM
(
    ENTRY_METHOD    VARCHAR2(250),
    PORT            NUMBER,
    SHIP_TYPE       VARCHRA2(250),
    SHIP_CAP        NUMBER,
    BILL_CODE       NUMBER
    BILL_DATE       DATE
);
```

At the group level there will be several views, such as:

```
V_CFA_PO_IMPORT
(
    ENTRY_METHOD    VARCHAR2(250),
    PORT            NUMBER
);
V_CFA_PO_SHIPPING

(
SHIP_TYPE VARCHRA2(250),
    SHIP_CAP        NUMBER
);
V_CFA_PO_BILLING

(
BILL_CODE NUMBER
    BILL_DATE       DATE
);
```

Both levels are optional and you can determine the suitable view level based on your business needs.

Additional Consideration

When group sets, groups, and attributes are defined for CFAS, the information is stored as metadata on a series of base RMS tables. When the database objects are created based on the metadata set up, they are created as generic tables, with column names like NUMBER_1 and VARCHAR2_4. Since this makes querying data for the attributes very difficult, views are created when the activation scripts are run at the group set and group level.

The attribute group set view will contain all the groups and attributes in that group set. The view at the group level will only contain the attributes for that particular group. Depending on the way you have organized the attributes or the particular needs of the query, one or the other may be used. For example, if an attribute group (A) is created with 4 attributes for Order, the view will look similar to the following:

- Order number
- <Attribute A1>
- <Attribute A2>
- <Attribute A3>
- <Attribute A4>

At the group set level, if there were two attribute groups in the set with group A being one and group B (with 5 attributes) the other, it will look similar to the following:

- Order number
- <Attribute A1>
- <Attribute A2>
- <Attribute A3>
- <Attribute A4>
- <Attribute B1>
- <Attribute B2>
- <Attribute B3>
- <Attribute B4>
- <Attribute B5>

Note: The group set and group numbers are not part of the views.

CFAS Staging Table

The CFAS staging table is used for importing data from external source to the extension tables (reverse functionality of the CFAS access views).

The name of the staging table is not stored in the CFAS installation table, but derived from value stored in the CFA_EXT_ENTITY.CUSTOM_EXT_TABLE by prefixing with *STG_*. A single staging table is created per extension entity.

EXT_ENTITY_ID	BASE_RMS_TABLE	CUSTOM_EXT_TABLE	Derived CFAS Staging Table Name
3000000001	ORDHEAD	ORDHEAD_CFA_EXT	STG_ORDHEAD_CFA_EXT
3000000002	ITEM_SUPP_COUNTRY	ITEM_SUP_CTRY _CFA_EXT	STG_ ITEM_SUP_CTRY _CFA_EXT
3000000003	STORE	STORE_CFA_EXT	STG_ STORE_CFA_EXT
3000000004	SUPS	SUPS_CFA_EXT	STG_ SUPS_CFA_EXT

The columns of the staging table will be all the values in the CFA_ATTRIB.VIEW_COL_NAME for the extended entity.

Additional Consideration

Staging tables are created for the attribute group set level when the activation scripts are run. The layout of the staging tables are the same as the layout of the view. Staging tables allow data from external sources to update the flex attributes. You can use the CFAS Load scripts to take the data into these staging tables and load them into the CFAS tables.

The data load scripts include only basic validation (for example, data type) for the data to be uploaded. Errors with the data (for example, invalid store number) will not be caught until the users open the relevant form containing the attributes. Ensure that you set up the relevant validation before loading the data from the staging table.